# Understanding and Mitigating Exploding Inverses in Invertible Neural Networks

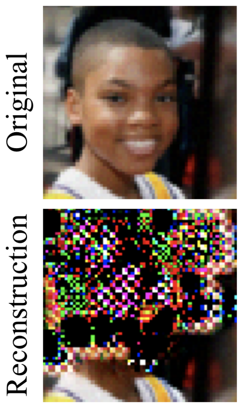
**Jens Behrmann***[1] **Paul Vicol***[2,3] **Kuan-Chieh Wang***[2,3] **Roger Grosse**[2,3] **Jörn-Henrik Jacobsen**[2,3]
[1]University of Bremen [2]University of Toronto [3]Vector Institute * Equal contribution

## Abstract

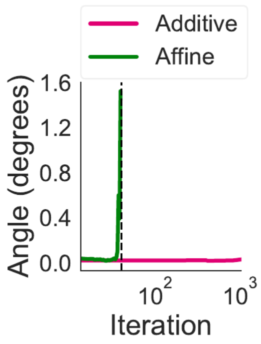
Invertible neural networks (INNs) have been used to design generative models, implement memory-saving gradient computation, and solve inverse problems. In this work, we show that commonly-used INN architectures suffer from exploding inverses and are thus prone to becoming numerically non-invertible. Across a wide range of INN use-cases, we reveal failures including the non-applicability of the change-of-variables formula on in- and out-of-distribution (OOD) data, incorrect gradients for memory-saving backprop, and the inability to sample from normalizing flow models. We further derive bi-Lipschitz properties of atomic building blocks of common architectures. These insights into the stability of INNs then provide ways forward to remedy these failures. For tasks where local invertibility is sufficient, like memory-saving backprop, we propose a flexible and efficient regularizer. For problems where global invertibility is necessary, such as applying normalizing flows on OOD data, we show the importance of designing stable INN building blocks.

## 1 INTRODUCTION

Invertible neural networks (INNs) have become a standard building block in the deep learning toolkit (Papamakarios et al., 2019; Donahue and Simonyan, 2019). Invertibility is useful for training normalizing flow (NF) models with exact likelihoods (Dinh et al., 2014, 2017), increasing posterior flexibility in VAEs (Rezende and Mohamed, 2015), learning transition operators in MCMC samplers (Song et al., 2017), computing memory-efficient gradients (Gomez et al., 2017), allowing for bi-directional training (Grover et al., 2018), solving inverse problems (Ardizzone et al., 2019a) and analyzing adversarial robustness (Jacobsen et al., 2019).



All the aforementioned applications rely on the assumption that the theoretical invertibility of INNs carries through to their numerical instantiation. In this work, we challenge this assumption by probing their inverse stability in generative and discriminative modeling settings. As a motivating example, on the left we show an image $x$ from within the dequantization distribution of a training example, and the reconstructed image $F^{-1}(F(x))$ from a competitive CelebA normalizing flow model $F$ (Kingma and Dhariwal, 2018). In the same vein as exploding gradients in RNNs, here the inverse mapping explodes, leading to severe reconstruction errors up to `Inf/NaN` values. The model exhibits similar failures both on out-of-distribution data and on samples from the model distribution (discussed in Section 4.1). Interestingly, none of these failures are immediately apparent during training. Hence, NFs can silently become non-invertible, violating the assumption underlying their main advantages—exact likelihood computation and efficient sampling (Papamakarios et al., 2019).



Memory-saving gradient computation (Gomez et al., 2017) is another popular application of INNs where exploding inverses can be detrimental. On the left, we show the angle between (1) gradients obtained from standard backprop and (2) memory-efficient gradients obtained using the inverse mapping to recompute

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

activations, during training of additive- and affine-coupling INN classifiers on CIFAR-10. The affine model exhibits exploding inverses, leading to a rapidly increasing angle that becomes `NaN` after the dashed vertical line—making memory-efficient training infeasible—whereas the additive model is stable. This highlights the importance of understanding the influence of different INN architectures on stability. Different tasks may have different stability requirements: NFs require the model to be invertible on training and test data and, for many applications, on out-of-distribution data as well. In contrast, memory-saving gradients only require the model to be invertible on the training data, to reliably compute gradients.

As we will show in our numerical experiments, the aforementioned failures are not a rare phenomenon, but are a concern across most application areas of INNs. To provide an understanding of these failures, we study the elementary components that influence the stability of INNs: 1) bi-Lipschitz properties of INN blocks (section 3.1); 2) the effect of the training objective (section 3.3); and 3) task-specific requirements for the inverse computations (section 4 on global invertibility for change-of-variables vs. local invertibility for accurate memory-efficient gradients). Finally, putting our theoretical analysis into practice, we empirically verify solutions to overcome exploding inverses. These solutions follow two main paradigms: 1) Enforcing global stability using Lipschitz-constrained INN architectures or 2) regularization to enforce local stability. In summary, we both uncover exploding inverses as an issue across most use-cases of INNs and provide ways to mitigate this instability. Our code can be found at: `http://www.github.com/asteroidhouse/INN-exploding-inverses`.

## 2 INVERTIBLE NETWORKS

Invertible neural networks (INNs) are bijective functions with a forward mapping $F : \mathbb{R}^d \to \mathbb{R}^d$ and an inverse mapping $F^{-1} : \mathbb{R}^d \to \mathbb{R}^d$. This inverse can be given in closed-form (Dinh et al., 2017; Kingma and Dhariwal, 2018) or approximated numerically (Behrmann et al., 2019; Song et al., 2019). Central to our analysis of the stability of INNs is bi-Lipschitz continuity:

**Definition 1** (Lipschitz and bi-Lipschitz continuity). *A function $F : \mathbb{R}^d \to \mathbb{R}^d$ is called* Lipschitz continuous *if there exists a constant $L := \mathrm{Lip}(F)$ such that:*

$$\|F(x_1) - F(x_2)\| \leq L\|x_1 - x_2\|, \quad \forall\, x_1, x_2 \in \mathbb{R}^d. \quad (1)$$

*If an inverse $F^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ and a constant $L^* := \mathrm{Lip}(F^{-1})$ exists such that for all $y_1, y_2 \in \mathbb{R}^d$*

$$\|F^{-1}(y_1) - F^{-1}(y_2)\| \leq L^*\|y_1 - y_2\| \quad (2)$$

*holds, then $F$ is called* bi-Lipschitz continuous. *Furthermore, $F$ or $F^{-1}$ is called* locally Lipschitz continuous *in $[a, b]^d$, if the above inequalities hold for $x_1, x_2$ or $y_1, y_2$ in the interval $[a, b]^d$.*

As deep-learning computations are carried out with limited precision, numerical error is always introduced in both the forward and inverse passes. Instability in either pass will aggravate this imprecision, and can make an *analytically* invertible network *numerically non-invertible.* If the singular values of the Jacobian of the inverse mapping can become arbitrarily large, we refer to this effect as an *exploding inverse.*

To obtain a better understanding in the context of INNs, we first define coupling blocks and then study numerical error and instability in a toy setting. Let $I_1, I_2$ denote disjoint index sets of $\{1, \dots, d\}$ to partition vectors $x \in \mathbb{R}^d$. For example, they denote a partition of feature channels in a convolutional architecture. Additive coupling blocks (Dinh et al., 2014) are defined as:

$$\begin{aligned} F(x)_{I_1} &= x_{I_1} \\ F(x)_{I_2} &= x_{I_2} + t(x_{I_1}) \end{aligned} \quad (3)$$

and affine coupling blocks (Dinh et al., 2017) as:

$$\begin{aligned} F(x)_{I_1} &= x_{I_1} \\ F(x)_{I_2} &= x_{I_2} \odot g(s(x_{I_1})) + t(x_{I_1}), \end{aligned} \quad (4)$$

where $t, s : \mathbb{R}^{|I_1|} \to \mathbb{R}^{|I_2|}$ are Lipschitz continuous and $\odot$ denotes elementwise multiplication. Hence, coupling blocks differ only in their scaling [1].

To understand how numerical error occurs in coupling blocks, consider the following example: consider $x \in \mathbb{R}^2$ and the trivial partition $I_1 = 1$ and $I_2 = 2$. Further, let $t(x_1) = 0.123456789$ and $g(s(x_1)) = 0.1$. Floating-point operations introduce rounding errors when numbers from different scales are summed up. Figure 1 visualizes the reconstruction error introduced at various depth for the given $t(x_1), g(s(x_1))$. In particular, the additive coupling INN has small numerical errors when reconstructing due to rounding errors in the summation. The affine blocks, however, show an exploding inverse effect since the singular values of the forward mapping tend to zero as depth increases. Thus, while being analytically invertible, the numerical errors and instability renders the network numerically non-invertible even in this simple example.

To formalize the connection of numerical errors and Lipschitz constants, consider $F(x) = z$ as the analytical

[1] Affine blocks scale the input with an elementwise function $g$ that has to be non-zero everywhere—common choices are sigmoid or $\exp(\cdot)$.

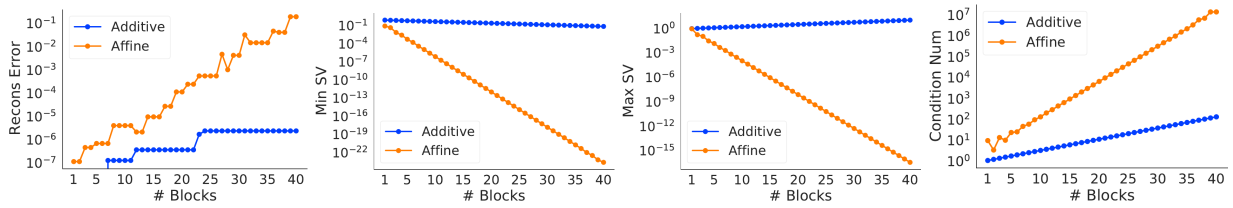


Figure 1: **Numerical error and instability in coupling blocks:** Consider a coupling block with $t(x_1) = 0.123456789$ and $g(s(x_1)) = 0.1$ (see Section 2). The input is $(x_1, x_2) = (1, 1)$. From left to right, we show the: 1) reconstruction error; 2) minimum singular value $\sigma_d$; 3) maximum singular value $\sigma_1$; and 4) condition number, $\sigma_1/\sigma_d$, as a function of the number of INN blocks.

exact forward computation and $F_\delta(x) = z + \delta =: z_\delta$ as the floating-point inexact computation with error $\delta$. In order to bound the error in the reconstruction due to the imprecision in the forward mapping, let $x_{\delta_1} = F^{-1}(z_\delta)$. Now consider:

$$\|x - x_{\delta_1}\|_2 \leq \text{Lip}(F^{-1})\|z - z_\delta\|_2 = \text{Lip}(F^{-1})\|\delta\|_2,$$

where $\text{Lip}(F^{-1})$ is used to bound the influence of the numerical error introduced in the forward mapping. Additionally, similarly to the forward mapping, the inverse mapping adds numerical imprecision. Thus, we introduce $F_\delta^{-1}(z_\delta) = x_{\delta_1} + \delta_2 := x_{\delta_2}$. Hence, we obtain the bound:

$$\begin{aligned}\|x - (x_{\delta_1} + \delta_2)\|_2 &\leq \|x - x_{\delta_1}\|_2 + \|\delta_2\|_2 \\ &\leq \text{Lip}(F^{-1})\|z - z_\delta\|_2 + \|\delta_2\|_2 \\ &= \text{Lip}(F^{-1})\|\delta\|_2 + \|\delta_2\|_2.\end{aligned}$$

While obtaining quantitative values for $\delta$ and $\delta_2$ for a model as complex as deep INNs is hard, the above formalization still provides insights into the role of the inverse stability when reconstructing inputs. In sum, INNs are designed to be analytically invertible, but numerical errors are always present. How much these errors are magnified is bounded by their Lipschitz continuity (stability).

## 3 STABILITY OF INVERTIBLE NEURAL NETWORKS

We first discuss bi-Lipschitz properties of common INN building blocks, and how certain architectures suffer from exploding inverses in Section 3.1. Then we explore how to stabilize INNs globally (Section 3.2) and locally (Section 3.3).

### 3.1 Lipschitz Properties of INN Blocks

Research on INNs has produced a large variety of architectural building blocks. Here, we build on the work of Behrmann et al. (2019), that proved bi-Lipschitz bounds for invertible ResNets. In particular, we derive Lipschitz bounds of coupling-based INNs and provide an overview of the stability of other common building blocks. Most importantly for our subsequent discussion are the qualitative insights we can draw from this analysis. Hence, we summarize these results in Theorem 2 and provide the quantitative Lipschitz bounds as lemmas in Appendix B.

We start with our main result on coupling blocks, which are the most commonly used INN architectures. While they only differ in the scaling, this difference strongly influences stability:

**Theorem 2** (Stability of additive and affine blocks)**.** *Consider additive and affine blocks as in Eq. 3 and Eq. 4 and assume the same function $t$. Further, let $t, s, g$ Lipschitz continuous, continuously differentiable and non-constant functions. Then the following differences w.r.t. bi-Lipschitz continuity hold:*

*(i) Affine blocks have strictly larger (local) bi-Lipschitz bounds than additive blocks.*

*(ii) There is a global bound on $\text{Lip}(F)$ and $\text{Lip}(F^{-1})$ for additive blocks, but only local bounds for affine blocks, i.e. there exist no $\text{Lip}(F) > 0$ and $\text{Lip}(F^{-1}) > 0$ such that Eqs. 1, 2 hold over $\mathbb{R}^d$.*

The proof is given in Appendix B.1, together with upper bounds in Lemmas 5 and 6. Note that an upper bound on $\text{Lip}(F)$ provides a lower bound on $\text{Lip}(F^{-1})$ and vice versa. These differences offer two main insights. First, affine blocks can have arbitrarily large singular values in the inverse Jacobian, i.e. an exploding inverse. Thus, they are more likely to become numerically non-invertible than additive blocks. Second, controlling stability in each architecture requires fundamentally different approaches since additive blocks have global bounds, while affine blocks are not globally bi-Lipschitz.

In addition to the Lipschitz bounds of coupling layers, we provide an overview of known Lipschitz bounds of other common INN building blocks in Table 3 (Appendix A). Besides coupling-based approaches we cover free-form approaches like Neural ODEs (Chen et al., 2018) and i-ResNets (Behrmann et al., 2019). Note that the bounds provide the worst-case stability and are primarily meant to serve as a guideline for the design of invertible blocks.

## 3.2 Controlling Global Stability of INNs

As we showed in the previous section, each INN building block has its own stability properties (see Table 3 for an overview). The bi-Lipschitz bounds of additive coupling blocks can be controlled using a similar strategy to i-ResNets. Via spectral normalization (Miyato et al., 2018), it is possible to control the Lipschitz-constant of $t$ in Eq. 3, which guarantees stability via the bounds from Lemma 5. On the other hand, spectral normalization does not provide guarantees for affine blocks, as they are not globally bi-Lipschitz over $\mathbb{R}^d$ due to dependence on the range of the inputs $x$ (see Theorem 2).

**Modified Affine Scaling.** A natural way to increase stability of affine blocks is to consider different elementwise scaling functions $g$, see Ardizzone et al. (2019b) for an example of such a modification. In particular, avoiding scaling by small values strongly influences the inverse Lipschitz bound (see Lemma 6, Appendix B). Thus, a modification guided by Lemma 6 would be to adapt the sigmoid scaling to output values in a restricted range such as $(0.5, 1)$ rather than the standard range $(0, 1)$. As we show in our experiments (Sections 4.1.1 & 4.1.2), this indeed improves stability. However, it may still suffer from exploding inverses as there is no global Lipschitz bound (see Theorem 2).

## 3.3 Controlling Local Stability of INNs

While the previous section aimed at controlling global stability, in this section we discuss how penalty functions and the training objective itself stabilize INNs locally, i.e. around inputs $x \in \mathbb{R}^d$.

### 3.3.1 Bi-Directional Finite Differences Regularization

Penalty terms on the Jacobian can be used to enforce local stability (Sokolić et al., 2017; Hoffman et al., 2019). Their connection to Lipschitz bounds can be understood using the identity from Federer (1969, Thm. 3.1.6): if $F : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous and differentiable, then we have:

$$\begin{aligned}\mathrm{Lip}(F) &= \sup_{x\in\mathbb{R}^d} \|J_F(x)\|_2 \\ &= \sup_{x\in\mathbb{R}^d} \sup_{\|v\|_2=1} \|J_F(x)v\|_2, \end{aligned} \tag{5}$$

where $J_F(x)$ is the Jacobian matrix of $F$ at $x$ and $\|J_F(x)\|_2$ denotes its spectral norm. To approximate the RHS of Eq. 5 from below, we obtain $v \in \mathbb{R}^d$ as $v/\|v\|_2$ with $v \sim \mathcal{N}(0, I)$, which uniformly samples from the unit sphere (Muller, 1959). For given $x$ and $v$, the term $\|J_F(x)v\|_2$ can be added to the loss function as a regularizer. However, training with such a penalty term requires double-backpropagtion (Drucker and Le Cun, 1992; Etmann, 2019), which makes recomputing the pre-activations during backprop via the inverse (Gomez et al., 2017) (memory-saving gradients) difficult. Thus, in addition to randomization, we introduce a second approximation using finite differences as:

$$\begin{aligned}&\sup_{x\in\mathbb{R}^d} \sup_{\|v\|_2=1} \|J_F(x)v\|_2 \approx \\ &\sup_{x\in\mathbb{R}^d} \sup_{\|v\|_2=1} \frac{1}{\varepsilon}\|F(x) - F(x+\varepsilon v)\|_2, \end{aligned} \tag{6}$$

with a step-size $\varepsilon > 0$. The approximation error can be estimated via Taylor expansion and has to be traded off with *catastrophic cancellation* due to subtracting near-identical float values (An et al., 2011). Since we aim at having both stable forward and inverse mappings, we employ this penalty on both directions $F$ and $F^{-1}$, and call it *bi-directional finite differences regularization* (abbreviated FD). Details about this architecture agnostic regularizer and its computational overhead are provided in Appendix G.

### 3.3.2 Influence of the Normalizing Flow Loss on Stability

In addition to the INN architecture and local regularization such as bi-directional FD introduced in Section 3.3.1, the training objective itself can impact local stability. Here, we examine the stabilization effect of the commonly used normalizing flow (NF) objective (Papamakarios et al., 2019). Consider a parametrized diffeomorphism $F_\theta : \mathbb{R}^d \to \mathbb{R}^d$ and a base distribution $p_Z$. By a change-of-variables, we have for all $x \in \mathbb{R}^d$

$$\log p_\theta(x) = \log p_Z(F_\theta(x)) + \log |\det J_{F_\theta}(x)| , \tag{7}$$

where $J_{F_\theta}(x)$ denotes the Jacobian of $F_\theta$ at $x$. The log-determinant in Eq. 7 can be expressed as:

$$\log |\det J_{F_\theta}(x)| = \sum_{i=1}^{d} \log \sigma_i(x), \tag{8}$$

where $\sigma_i(x)$ denotes the $i$-th singular value of $J_{F_\theta}(x)$. Thus, minimizing the negative log-likelihood as $\min_\theta -\log p_\theta(x)$ involves maximizing the sum of the log singular values (Eq. 8). Due to the slope of the logarithmic function $\log(x)$, very small singular values are avoided more strongly than large singular values are favored. Thus, the inverse of $F_\theta$ is encouraged to be more stable than the forward mapping. Furthermore when using $Z \sim \mathcal{N}(0, I)$ as the base distribution, we

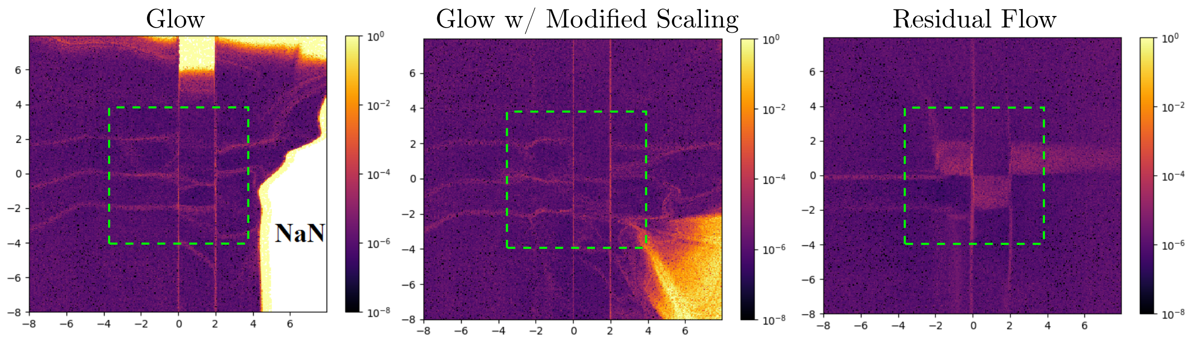


Figure 2: **Reconstruction error on 2D checkerboard data. Left:** an affine model with standard sigmoid scaling in $(0, 1)$; **Middle:** a more stable affine model with scaling in $(0.5, 1)$; **Right:** a Residual Flow model (Chen et al., 2019). The green boxes highlight the training data distribution $[-4, 4]$; we see that both affine models become unstable outside this distribution, while the Residual Flow remains stable.

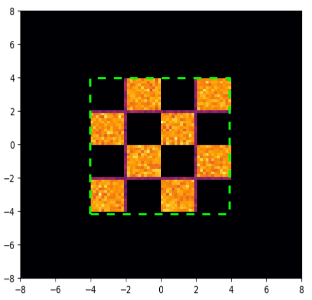


Figure 3: **Samples from 2D checkerboard.**

minimize:

$$-\log p_Z(F_\theta(x)) \propto \|F_\theta(x)\|_2^2,$$

which bounds the $\ell_2$-norm of the outputs of $F_\theta$. Due to this effect, large singular values are avoided and the mapping $F_\theta$ is further locally stabilized.

Thus, the two terms of the normalizing flow objective have complementary effects on stability: the log-determinant increases all singular values, but has a stronger effect on small singular values than on large ones, improving inverse stability, while the base term encourages the output of the function to have small magnitude, improving forward stability. If additional stability is required, bounding the $\ell_2$-norm of intermediate activations would further avoid fluctuations, where a subflow exhibits high magnitude that is cancelled out by the subsequent subflow. The effect of the NF objective, however, acts only on the training data $x \in \mathbb{R}^d$ and is thus not able to globally stabilize INNs, as we show in our experiments (Section 4.1).

## 4 EXPERIMENTS

First, we show that exploding inverses are a concern across most application areas of INNs. Second, we aim to provide ways to mitigate instability. For this we conduct experiments on two tasks: generative modeling with normalizing flows (Section 4.1) and memory-efficient gradient computation for supervised learning (Section 4.2). Due to the growing body of INN architectures we had to restrict our experimental study to a subset of INNs: additive/ affine coupling blocks (Dinh et al., 2014, 2017) and Residual Flows (Chen et al., 2019). This particular choice was guided by the simplicity of coupling blocks and by the close link of Residual Flows to stability.

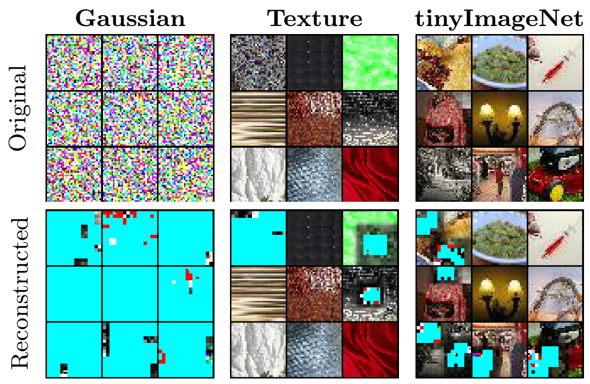
### 4.1 Non-Invertibility in Normalizing Flows

Here we show that INNs can become numerically non-invertible even when trained with the normalizing flow (NF) loss (despite encouraging local stability, see Section 3.3.2). We study this behavior on out-of-distribution data and identify the exploding inverse effect in the data and model distribution.

#### 4.1.1 Instability on Out-of-Distribution Data

Because NFs allow for efficient likelihood computation, they have been used in several likelihood-based approaches for out-of-distribution (OOD) detection (Nalisnick et al., 2019; Fetaya et al., 2020). Here we show, however, that certain classes of flows can be numerically non-invertible on OOD data, implying that the likelihoods computed on such data are not meaningful because the change-of-variables formula no longer applies.

**2D Checkerboard.** First we consider a 2D checkerboard distribution (see samples in Figure 3 and further madetails in Appendix C). Despite being ill-posed due to the discontinuous density (jumps at the edges of the checkerboard), it is often used as a benchmark for NFs (Chen et al., 2019; Grathwohl et al., 2019). The discontinuity of the dataset is manifested in two main ways in Figure 2: 1) at these jumps, the model becomes unstable, which leads to larger reconstruction errors (see slightly expressed grid-like pattern), 2) affine models can become non-invertible outside the data domain. Figure 2 further shows reconstruction errors from a modified affine model—which is more stable, but still



| | Glow | | ResFlow | |
|---|---|---|---|---|
| **Dataset** | **% Inf** | **Err** | **% Inf** | **Err** |
| CIFAR-10 | 0 | 6.3e-5 | 0 | 2.9e-2 |
| Uniform | 100 | - | 0 | 1.7e-2 |
| Gaussian | 100 | - | 0 | 7.2e-3 |
| Rademacher | 100 | - | 0 | 1.9e-3 |
| SVHN | 0 | 5.5e-5 | 0 | 7.3e-2 |
| Texture | 37.0 | 7.8e-2 | 0 | 2.0e-2 |
| Places | 24.9 | 9.9e-2 | 0 | 2.9e-2 |
| tinyImageNet | 38.9 | 1.6e-1 | 0 | 3.5e-2 |

Figure 4: **Left:** Reconstructions of OOD data, using a CIFAR-10 pre-trained Glow model. Broken regions (`NaN` or `Inf`) in the reconstructions are plotted in cyan. **Right:** Mean $\ell_2$ reconstruction errors on in-distribution (CIFAR-10) and out-of-distribution data, for a pre-trained Glow and Residual Flow. We used three synthetic noise datasets {Uniform, Gaussian, Rademacher} as well as SVHN (Netzer et al., 2011), Texture (Cimpoi et al., 2014), Places (Zhou et al., 2017), and tinyImageNet. We used 10,000 samples from each OOD dataset, and we report 1) the percentage of images that yielded `Inf` reconstruction error; and 2) the mean reconstruction error for the non-`Inf` samples, see `Err` column.

| | Additive | Affine | Mod. Affine |
|---|---|---|---|
| Data BPD (Train) | 3.29 | 3.27 | 3.25 |
| Data BPD (Test) | 3.55 | 3.51 | 3.5 |
| Reliable Sample BPD | ✓ | ✗ | ✓ |
| No Visible Sample Recon Err. | ✓ | ✗ | ✓ |

Table 1: **Flow results on CIFAR10.** Bits-per-dimension (BPDs) reported at 100k updates. The bottom 2 rows show whether problems occur during training. The affine model was unstable w.r.t. model samples, and the additive and modified affine models are stable.

suffers from exploding inverses in OOD areas—and a Residual Flow (Chen et al., 2019), which has low reconstruction error globally, consistent with our stability analysis.

**CIFAR-10 OOD.** Next, we evaluated CIFAR-10 pre-trained Glow and Residual Flow models on a set of OOD datasets from Hendrycks and Gimpel (2016); Liang et al. (2017); Nalisnick et al. (2019). [2] Figure 4 shows qualitative reconstruction errors for Glow on three OOD datasets, as well as the numerical reconstruction errors for Glow and ResFlow models on all OOD datasets. While the Residual Flow was always stably invertible, Glow was non-invertible on all datasets except SVHN. This indicates that OOD detection methods based on Glow likelihoods may be unreliable. In Appendix D, we provide additional details, as well as a comparison of the stability of additive and affine coupling models on OOD data during training.

[2]For Glow and Residual Flows, we used the pre-trained models from `https://github.com/y0ast/Glow-PyTorch` and `https://github.com/rtqichen/residual-flows`, respectively.

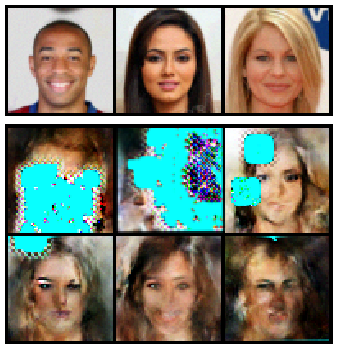


Figure 5: **Instability in model-distribution.** The top row shows data from CelebA64 and the two bottom rows show samples from an affine model during training at epoch 20. `NaN` pixels are visualized in cyan.

### 4.1.2 Instability in the Data Distribution and further Failures

In this section, we further identify failures within the model and data distributions. In particular, we study the stability of the inverse on the model distribution by sampling $z$ from the base distribution (rather than obtaining $z$ via a forward pass on some data). For this, we trained both additive/affine models on CIFAR-10, and an affine model on CelebA64 (see details in Appendix D). We report quantitative results during training on CIFAR-10 in Table 1. The affine model was unstable w.r.t. model samples. Furthermore in Figure 5, we show samples from the affine CelebA64 model, which has `NaN` values in multiple samples.

**Non-Invertible Inputs within the Dequantization Region.** We can further expose non-invertibility even in the data distribution. By optimizing within

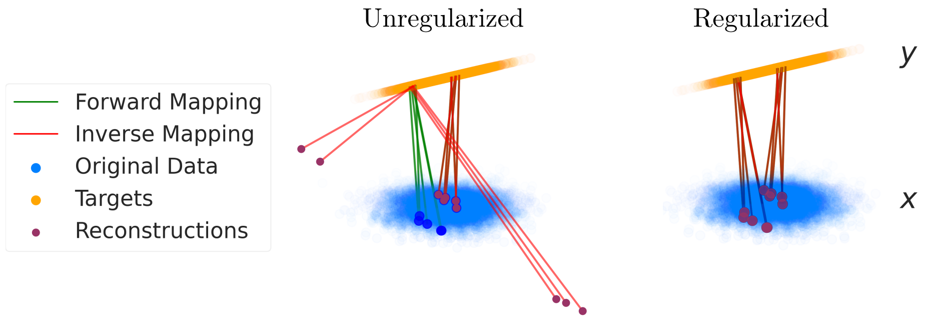


Figure 6: **Exploding inverses on a 2D regression task.** A Glow model is trained to map between two 2D Gaussian distributions $(x_1, x_2) \rightarrow (y_1, y_2)$, where $y_2$ has low variance, so that we are essentially mapping from 2D space onto a 1D subspace. **Left:** An unregularized model exhibits exploding inverses, illustrated by the points that are mapped far outside the original data distribution by the inverse mapping. **Right:** Regularizing the model by adding the normalizing flow objective with a small coefficient (1e-8) stabilizes the mapping.

the dequantization distribution of a datapoint we are able to find regions that are poorly reconstructed by the model. Note that the inputs found this way are valid samples from the training data distribution. See Appendix E for details.

**Instability beyond the NF loss.** In Appendix H we provide results on bi-directional training with INNs in the Flow-GAN setting (Grover et al., 2018). While additive models did not show instabilities thus far, they exhibit exploding inverses when trained solely adversarially. Lastly, this exploration demonstrates how our analysis can be leveraged to choose appropriate tools for stabilizing INNs, and in this case improve model samples while retaining competitive bits-per-dimension (BPDs).

## 4.2 Supervised Learning with Memory Efficient Gradients

For supervised learning, INNs enable memory-efficient training by re-computing intermediate activations in the backward pass, rather than storing them in memory during the forward pass (Gomez et al., 2017). This enables efficient large-scale generative modeling (Donahue and Simonyan, 2019) and high-resolution medical image analysis (Etmann et al., 2020). Re-computing the activations during the backward pass, however, relies on the inverse being numerically precise locally around the training data. This is a weaker requirement than the global stability we desire for NFs such that they can be applied to OOD data. However, in this section we show that even this weaker requirement can be violated, and how to mitigate these failures by local regularization as discussed in Section 3.3.

**Toy 2D Regression.** In contrast to NFs, where the likelihood objective encourages local stability, there is no default mechanism to avoid unstable inverses in supervised learning (e.g., classification or regression). As an example, consider a simple 2D regression problem visualized in Figure 6. Here, the targets $y$ lie almost on a 1D subspace, which requires the learned mapping to contract from 2D to 1D. Even in such a simple task, a Glow regression model becomes non-invertible, as shown by the misplaced reconstructions for the unregularized model. Additional details are provided in Appendix F. This illustrates the importance of adding regularization terms to supervised objectives to maintain invertibility, as we do next for memory-efficient training on CIFAR-10.

**CIFAR-10 Classification.** Here we show that INN classifiers on CIFAR-10 can become non-invertible—making it impossible to compute accurate memory-saving gradients—and that local regularization by adding either the finite differences (FD) penalty or the normalizing flow (NF) objective with a small coefficient stabilizes these models, enabling memory-efficient training. We focused on additive and affine models with architectures similar to Glow (Kingma and Dhariwal, 2018), with either $1 \times 1$ convolutions or shuffle permutations and ActNorm between building blocks. We used only coupling approaches, because i-ResNets (Behrmann et al., 2019) are not suited for memory-efficient gradients due to their use of an expensive iterative inverse. Experimental details and extended results are provided in Appendix G.

In Table 2, we compare the performance and stability properties of unregularized additive and affine models, as well as regularized versions using either FD or NF penalties (Appendix G shows the effects of different regularization strengths). Note that we do not aim to achieve SOTA accuracy, and these accuracies match those reported for a similar Glow classifier by Behrmann et al. (2019). In particular, we observe how affine models suffer from exploding inverses and are thus not suited for computing memory-efficient gradients. Both the NF and FD regularizers mitigate the instability, yielding similar test accuracies to the unregularized model, while maintaining small reconstruc-

| Model | Regularizer | Inv? | Test Acc | Recons. Err. | Cond. Num. | Min SV | Max SV |
|---|---|---|---|---|---|---|---|
| Additive Conv | None | ✓ | 89.73 | 4.3e-2 | 7.2e+4 | 6.1e-2 | 4.4e+3 |
| | FD | ✓ | 89.71 | 1.1e-3 | 3.0e+2 | 8.7e-2 | 2.6e+1 |
| | NF | ✓ | 89.52 | 9.9e-4 | 1.7e+3 | 3.9e-2 | 6.6e+1 |
| Affine Conv | None | ✗ | 89.07 | `Inf` | 8.6e14 | 1.9e-12 | 1.7e+3 |
| | FD | ✓ | 89.47 | 9.6e-4 | 1.6e+2 | 9.6e-2 | 1.5e+1 |
| | NF | ✓ | 89.71 | 1.3e-3 | 2.2e+3 | 3.5e-2 | 7.7e+1 |

Table 2: **Effect of the finite differences (FD) and normalizing flow (NF) regularizers** when training additive and affine INN Glow architectures using $1 \times 1$ convolutions for CIFAR-10 classification. For each setting, we report the test accuracy, the numerical reconstruction error, and the condition number and min/max singular values (SVs) of the Jacobian of the forward mapping. While the additive model was always stable, the unregularized affine model became highly unstable, with `Inf` reconstruction error; we observe that instability arises from the inverse mapping, as the min SV is 1.9e-12.

tion errors and condition numbers.[3] In Appendix G, we show that regularization keeps the angle between the true gradient and memory-saving gradient small throughout training. The computational overhead of the FD regularizer is only 1.26× that of unregularized training (Table 11 in App. G). We also experimented with applying the FD regularizer only to the inverse mapping, and while this can also stabilize the inverse and achieve similar accuracies, we found bi-directional FD to be more reliable across architectures. Applying regularization once every 10 iterations, bi-FD is only 1.06× slower than inverse-FD. As the FD regularizer is architecture agnostic and conceptually simple, we envision a wide-spread use for memory-efficient training.

## 5 RELATED WORK

**Invertibility and Stability of Deep Networks.** The inversion from activations in standard neural networks to inputs has been studied via optimization in input space (Mahendran and Vedaldi, 2014) or by linking invertibility and inverse stability for ReLU-networks (Behrmann et al., 2018). However, few works have studied the stability of INNs: Gomez et al. (2017) examined the numerical errors in the gradient computation when using memory-efficient backprop. Similarly to our empirical analysis, Jacobsen et al. (2018) computed the SVD of the Jacobian of an i-RevNet and found it to be ill-conditioned. Furthermore, i-ResNets (Behrmann et al., 2019) yield bi-Lipschitz bounds by design. Lastly, Chang et al. (2018) studied the stability of reversible INN dynamics in a continuous framework. In contrast, the stability of neural networks has been of major interest due to the problem of exploding/vanishing gradients as well as for training Wasserstein GANs (Arjovsky et al., 2017). Furthermore, adversarial examples (Szegedy et al., 2013) are strongly tied to stability and inspired our invertibility attack (Section 4.1.2).

[3]Note that the training with FD was performed memory-efficiently, while the None and NF settings were trained using standard backprop.

**Improving Stability of Invertible Networks.** Instability in INNs has been noticed in other works, yet without a detailed treatment. For example, Putzky and Welling (2019) proposed to employ orthogonal $1\times1$ convolutions to obtain accurate memory-efficient gradients and Etmann et al. (2020) used weight normalization for stabilization. Our finite differences regularizer, on the other hand, is architecture agnostic and could be used in the settings above. Furthermore, Ardizzone et al. (2019b) considered modified scaling in affine models to improve their stability. Neural ODEs (Chen et al., 2018) are another way to design INNs, and research their stability. Finlay et al. (2020); Yan et al. (2020); Massaroli et al. (2020) provide further insights into designing principled and stable INNs.

**Fixed-Point Arithmetic in INNs.** Maclaurin et al. (2015); MacKay et al. (2018) implement invertible computations using fixed-point numbers, with custom schemes to store information that is lost when bits are shifted, enabling exact invertibility at the cost of memory usage. As Gomez et al. (2017) point out, this allows exact numerical inversion when using additive coupling independent of stability. However, our stability analysis aims for a broadly applicable methodology beyond the special case of additive coupling.

**Invertible Building Blocks.** Besides the invertible building blocks listed in Table 3 (Appendix A), several other approaches like in Karami et al. (2019) have been proposed. Most prominently, autogressive models like IAF (Kingma et al., 2016), MAF (Papamakarios et al., 2017) or NAF (Huang et al., 2018) provide invertible models that are not studied in our analysis. Furthermore, several newer coupling layers that require numerical inversion have been introduced by Ho et al. (2019); Jaini et al. (2019); Durkan et al. (2019). In addition to the coupling-based approaches, multiple approaches (Chen et al., 2018; Behrmann et al., 2019; Chen et al., 2019; Grathwohl et al., 2019; Song et al., 2019) use numerical inversion schemes, where the interplay of numerical error due to stability and error due to the approximation of the inverse adds another dimension to the study of invertibility.

**Stability-Expressivity Tradeoff.** While Lipschitz constrained INNs like i-ResNets (Behrmann et al., 2019) allow to overcome many failures we observed with affine coupling blocks (Dinh et al., 2017), this constrains the flexibility of INNs. Hence, there is a tradeoff between stability and expressivity as studied in Jaini et al. (2020); Cornish et al. (2020). While numerical invertibility is necessary for a safe usage of INNs, mixtures of Lipschitz constrained INNs could be used for improved flexibility as suggested in Cornish et al. (2020).

## 6 CONCLUSION

Invertible Neural Networks (INNs) are an increasingly popular component of the modern deep learning toolkit. However, if analytical invertibility does not carry through to the numerical computation, their underlying assumptions break. When applying INNs, it is important to consider how the inverse is used. If local stability is sufficient, like for memory-efficient gradients, our finite difference penalty is sufficient as an architecture agnostic stabilizer. For global stability requirements e.g. when using INNs as normalizing flows, the focus should be on architectures that enable stable mappings like Residual Flows (Chen et al., 2019). Altogether we have shown that studying stability properties of both forward and inverse is a key step towards a complete understanding of INNs.

## Acknowledgements

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute `www.vectorinstitute.ai/#partners`. Jens Behrmann acknowledges the support by the Deutsche Forschungsgemeinschaft (DFG) within the framework of GRK 2224/1 "$\pi^3$: Parameter Identification – Analysis, Algorithms, Applications". Paul Vicol was supported by a JP Morgan AI Fellowship. We also would like to thank Joost van Amersfoort for insightful discussions, and members on the Vector's Ops team (George Mihaiescu, Relu Patrascu, and Xin Li) for their computation/infrastructure support.

### References

Heng-Bin An, Ju Wen, and Tao Feng. On finite difference approximation of a matrix-vector product in the Jacobian-free Newton–Krylov method. *Journal of Computational and Applied Mathematics*, 236(6): 1399 – 1409, 2011.

Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference 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.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

Uri M. Ascher. *Numerical methods for evolutionary differential equations*. Computational Science and Engineering. Society for Industrial and Applied Mathematics, 2008.

Jens Behrmann, Sören Dittmer, Pascal Fernsel, and Peter Maaß. Analysis of invariance and robustness via invertibility of ReLU-networks. *arXiv preprint arXiv:1806.09730*, 2018.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, 2019.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, 2019.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Conference on Computer Vision and Pattern Recognition*, 2014.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International Conference on Machine Learning*, 2020.

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

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, 2019.

Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

Conor Durkan, Artur Bekasov, Ian Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, 2019.

Christian Etmann. A closer look at double backpropagation. *arXiv preprint arXiv:1906.06637*, 2019.

Christian Etmann, Rihuan Ke, and Carola-Bibiane Schönlieb. iUNets: Fully invertible U-Nets with learnable up- and downsampling. *arXiv preprint arXiv:2005.05220*, 2020.

Herbert Federer. *Geometric Measure Theory*. Grundlehren der Mathematischen Wissenschaften. Springer, 1969.

Ethan Fetaya, Jörn-Henrik Jacobsen, Will Grathwohl, and Richard Zemel. Understanding the limitations of conditional generative models. In *International Conference on Learning Representations*, 2020.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ODE: The world of Jacobian and kinetic regularization. *arXiv preprint arXiv:2002.02798*, 2020.

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 2214–2224, 2017.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, 2019.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with Jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, 2018.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *International Conference on Learning Representations*, 2018.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *International Conference on Learning Representations*, 2019.

Priyank Jaini, Kira A. Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *International Conference on Machine Learning*, 2019.

Priyank Jaini, Ivan Kobyzev, Yaoliang Yu, and Marcus Brubaker. Tails of Lipschitz triangular flows. In *International Conference on Machine Learning*, 2020.

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

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

Durk 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.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger B Grosse. Reversible recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 9029–9040, 2018.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Conference on Computer Vision and Pattern Recognition*, 2014.

Stefano Massaroli, Michael Poli, Michelangelo Bin, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Stable neural flows. *arXiv preprint arXiv:2003.08063*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, 1959.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

Patrick Putzky and Max Welling. Invert to learn to invert. In *Advances in Neural Information Processing Systems*, 2019.

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

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-NICE-MC: Adversarial training for MCMC. In *Advances in Neural Information Processing Systems*, pages 5140–5150, 2017.

Yang Song, Chenlin Meng, and Stefano Ermon. Mint-Net: Building invertible neural networks with masked convolutions. In *Advances in Neural Information Processing Systems*, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Hanshu Yan, Jiawei Du, Vincent Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. In *International Conference on Learning Representations*, 2020.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*, 2019.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.