Understanding and Mitigating Exploding Inverses in Invertible Neural Networks SUPPLEMENTARY MATERIAL

- In Section A we provide a table of Lipschitz bounds for various INN building blocks.
- In Section B we provide statements on Lipschitz bounds and their corresponding proofs.
- In Section C we provide details for the 2D checkerboard experiments.
- In Section D we provide details for the OOD and sample evaluation experiments.
- In Section E we expose non-invertibility in the data distribution by optimizing within the dequantization distribution of a datapoint to find regions that are poorly reconstructed by the model.
- In Section F we provide details for the 2D toy regression experiment.
- In section G we provide details for the CIFAR-10 classification experiments.
- In Section H we discuss an outlook on bi-directional training with FlowGANs.

A TABLE OF LIPSCHITZ BOUNDS

Building Block	Forward Operation	Lipschitz Forward	Lipschitz Inverse
Additive Coupling Block (Dinh et al., 2014)	$F(x)_{I_1} = x_{I_1}$ $F(x)_{I_2} = x_{I_2} + t(x_{I_1})$	$\leq 1 + \mathrm{Lip}(t)$	$\leq 1 + \mathrm{Lip}(t)$
Affine Coupling Block (Dinh et al., 2017)	$F(x)_{I_1} = x_{I_1}$ $F(x)_{I_2} = x_{I_2} \odot g(s(x_{I_1})) + t(x_{I_1})$ $g(\cdot) \neq 0$	$\leq \max(1, c_g) + M$ local for $x \in [a, b]^d$ $g(x) \leq c_g$	$\leq \max(1, c_{\frac{1}{g}}) + M^*$ local for $y \in [a^*, b^*]^d$ $\frac{1}{g}(y) \leq c_{\frac{1}{g}}$
Invertible Residual Layer (Behrmann et al., 2019)	$F(x) = x + g(x)$ $\mathrm{Lip}(g) < 1$	$\leq 1 + \mathrm{Lip}(g)$	$\leq \frac{1}{1-\mathrm{Lip}(g)}$
Neural ODE (Chen et al., 2018)	$\frac{dx(t)}{dt} = F(x(t), t)$ $t \in [0, T]$	$\leq e^{\mathrm{Lip}(F)\cdot t}$	$\leq e^{\mathrm{Lip}(F)\cdot t}$
Diagonal Scaling (Dinh et al., 2014) **ActNorm** (Kingma and Dhariwal, 2018)	$F(x) = Dx$ D diagonal $D_{ii} \neq 0$	$= \max_i \lvert D_{ii}\rvert$	$= \frac{1}{\min_i \lvert D_{ii}\rvert}$
Invertible 1×1 Convolution (Kingma and Dhariwal, 2018)	$F(x) = PL(U + \mathrm{diag}(s)) =: W$ P permutation, L lower-triangular U upper-triangular, $s \in \mathbb{R}^d$	$\leq \lVert W\rVert_2$	$\leq \lVert W^{-1}\rVert_2$

Table 3: **Lipschitz bounds on building blocks of invertible neural networks.** The second column shows the operations of the forward mapping and the last two columns show bounds on the Lipschitz constant of the forward and inverse mapping. M in the row for the forward mapping of an affine block is defined as $M = \max(|a|, |b|) \cdot c_{g'} \cdot \mathrm{Lip}(s) + \mathrm{Lip}(t)$. Furthermore, M^* for the inverse of an affine block is $M^* = \max(|a^*|, |b^*|) \cdot c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) + c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) \cdot c_t + c_{\frac{1}{g}} \cdot \mathrm{Lip}(t)$. Note that the bounds of the affine blocks hold only locally.

B STATEMENTS ON LIPSCHITZ BOUNDS AND PROOFS

In this section, we provide our analysis of bi-Lipschitz bounds of common INN architectures. The obtained bounds are summarized in Table 3. In general, Lipschitz bounds for deep neural networks often tend to be loose in practice, because a derived bound for a single layer needs to be multiplied by the number of layers of then entire network. Thus these bounds are rarely used quantitatively, however such an analysis can reveal crucial qualitative differences between architecture designs. This is why, we provide the technical analysis of the bi-Lipschitz bounds in the appendix and discuss their qualitative implications in Section 3 of the main body.

The bounds for i-ResNets are taken from (Behrmann et al., 2019). For Neural ODEs (Chen et al., 2018), one needs to consider a Lipschitz constant $\mathrm{Lip}(F)$ that holds for all $t \in [0, T]$, i.e.

$$\|F(t, x_1) - F(t, x_2)\|_2 \leq \mathrm{Lip}(F)\|x_1 - x_2\|_2, \quad \text{for all} \quad t \in [0, T].$$

Then, the claimed bound is a standard result, see e.g. (Ascher, 2008, Theorem 2.3). Note that the inverse is given by $\frac{dy(t)}{dt} = -F(y(t), t)$, hence the same bound holds.

In the following, we proceed as follows: First, we state bi-Lipschitz bounds of additive coupling blocks as a lemma and proove them (Lemma 5). Their derivation is generally straightforward but somewhat technical at stages due to the handling of the partition. Second, we perform the same analysis for affine coupling blocks (Lemma 6). Third, we use these technical lemmas to proof Theorem 2 from the main body of the paper.

Before deriving the upper bounds, we note that the upper bounds on the bi-Lipschitz constants also provide lower bounds:

Remark 3 (Lower bounds via upper bounds)**.** *By reversing, upper bounds on the Lipschitz constant of the inverse mapping yield lower bounds on the Lipschitz constant of the forward and vice versa. This holds due to the following derivation: Let $x, x^* \in \mathbb{R}^d$ and $F^{-1}(z) = x$, $F^{-1}(z^*) = x^*$. By employing the definition of the forward and inverse Lipschitz constants, we have:*

$$\begin{aligned} \|x - x^*\| = \|F^{-1}(z) - F^{-1}(z^*)\| &\leq \mathrm{Lip}(F^{-1})\|z - z^*\| = \mathrm{Lip}(F^{-1})\|F(x) - F(x^*)\| \\ &\leq \mathrm{Lip}(F^{-1})\mathrm{Lip}(F)\|x - x^*\| \\ \Leftrightarrow \frac{1}{\mathrm{Lip}(F^{-1})}\|x - x^*\| \leq \|F(x) - F(x^*)\| &\leq \mathrm{Lip}(F)\|x - x^*\|. \end{aligned} \tag{9}$$

By denoting the upper bounds as $L \geq \mathrm{Lip}(F)$ and $L^* \geq \mathrm{Lip}(F^{-1})$ and using the same reasoning as above, we thus have:

$$L \geq \mathrm{Lip}(F) \geq \frac{1}{\mathrm{Lip}(F^{-1})} \geq \frac{1}{L^*} \quad \text{and} \quad L^* \geq \mathrm{Lip}(F^{-1}) \geq \frac{1}{\mathrm{Lip}(F)} \geq \frac{1}{L}.$$

Hence our bounds provide a rare case, where not only upper bounds on the Lipschitz constants of neural networks are known, but also lower bounds due to the bi-Lipschitz continuity.

Remark 4 (bi-Lipschitz constant)**.** *By considering inequality 9, it is further possible to introduce a single constant as $\mathrm{biLip}(F) = \max\{\mathrm{Lip}(F), \mathrm{Lip}(F^{-1})\}$ for which:*

$$\frac{1}{\mathrm{biLip}(F)}\|x - x^*\| \leq \|F(x) - F(x^*)\| \leq \mathrm{biLip}(F)\|x - x^*\|$$

holds. This constant is usually called the bi-Lipschitz constant. We, on the other hand, refer to both constants $\mathrm{Lip}(F), \mathrm{Lip}(F^{-1})$ as bi-Lipschitz constants. We use this slightly more descriptive language, because we are particularly interested in the stability of each mapping direction.

Now we consider coupling blocks and provide upper Lipschitz bounds:

Lemma 5 (Lipschitz bounds for additive coupling block)**.** *Let I_1, I_2 be a disjoint index sets of $\{1, ..., d\}$ and let I_1, I_2 be non-empty. Consider an additive coupling block (Dinh et al., 2014) as:*

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

where $t : \mathbb{R}^{|I_1|} \to \mathbb{R}^{|I_2|}$ is a Lipschitz continuous and differentiable function. Then, the Lipschitz constant of the forward mapping F and inverse mapping F^{-1} can be upper-bounded by:

$$\begin{aligned}\mathrm{Lip}(F) &\leq 1 + \mathrm{Lip}(t)\\ \mathrm{Lip}(F^{-1}) &\leq 1 + \mathrm{Lip}(t).\end{aligned}$$

Proof. To prove the Lipschitz bounds, we use the identity:

$$\mathrm{Lip}(F) = \sup_{x \in \mathbb{R}^d} \|J_F(x)\|_2.$$

Thus, in order to obtain a bound on the Lipschitz constant, we look into the structure of the Jacobian. Without loss of generality we consider $I_1 = \{1, \dots, m\}$ and $I_2 = \{m+1, \dots, d\}$, where $1 < m < d$. The general case, where I_1, I_2 are arbitrary disjoint and non-empty index sets, can be recovered by a permutation. A permutation is norm-preserving and thus does not influence the bound on the Lipschitz constant.

The Jacobian of F has a lower-block structure with an identity diagonal, i.e.

$$J_F(x) = \begin{pmatrix} I_{m\times m} & 0_{(m)\times(d-m)} \\ J_t(x) & I_{(d-m)\times(d-m)} \end{pmatrix},$$

where $J_t(x) \in \mathbb{R}^{(d-m)\times m}$ denotes the Jacobian of t at x. By using this structure, we can derive the following upper bound:

$$\begin{aligned}
\mathrm{Lip}(F)^2 &= \sup_{x\in\mathbb{R}^d} \|J_F(x)\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \|J_F(x)x^*\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \|(J_F(x)x^*)_{I_1}\|_2^2 + \|(J_F(x)x^*)_{I_2}\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \|x^*_{I_1}\|_2^2 + \|x^*_{I_2} + J_t(x)x^*_{I_1}\|_2^2 \\
&\leq \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \|x^*_{I_1}\|_2^2 + (\|x^*_{I_2}\|_2 + \|J_t(x)x^*_{I_1}\|_2)^2 \qquad (10)\\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \|x^*_{I_1}\|_2^2 + \|x^*_{I_2}\|_2^2 + 2\|x^*_{I_2}\|_2\|J_t(x)x^*_{I_1}\|_2 + \|J_t(x)x^*_{I_1}\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \|x^*\|_2^2 + 2\|x^*_{I_2}\|_2\|J_t(x)x^*_{I_1}\|_2 + \|J_t(x)x^*_{I_1}\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} 1 + 2\|x^*_{I_2}\|_2\|J_t(x)x^*_{I_1}\|_2 + \|J_t(x)x^*_{I_1}\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} 1 + 2\|J_t(x)x^*_{I_1}\|_2 + \|J_t(x)x^*_{I_1}\|_2^2 \\
&= \sup_{x\in\mathbb{R}^d} \sup_{\|x^*\|_2=1} \left(1 + \|J_t(x)x^*_{I_1}\|_2\right)^2 \\
&= \sup_{x\in\mathbb{R}^d} \left(1 + \|J_t(x)\|_2\right)^2 \\
\Rightarrow \mathrm{Lip}(F) &\leq 1 + \mathrm{Lip}(t).
\end{aligned}$$

Furthermore, the inverse of F can be obtained via the simple algebraic transformation ($y := F(x)$)

$$\begin{aligned}F^{-1}(y)_{I_1} &= y_{I_1}\\ F^{-1}(y)_{I_2} &= y_{I_2} - t(y_{I_1}).\end{aligned}$$

Since the only difference to the forward mapping is the minus sign, the Lipschitz bound for the inverse is the same as for the forward mapping. □

Lemma 6 (Lipschitz bounds for affine coupling block)**.** *Let I_1, I_2 be a disjoint partition of indices $\{1, ..., d\}$ and let I_1, I_2 be non-empty. Consider an affine coupling block (Dinh et al., 2017) as:*

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

where $g, s, t : \mathbb{R}^{|I_1|} \to \mathbb{R}^{|I_2|}$ are Lipschitz continuous, continuously differentiable and non-constant functions. Then, the Lipschitz constant of the forward F can be locally bounded for $x \in [a, b]^d$ as:

$$\mathrm{Lip}(F) \leq \max(1, c_g) + M,$$

where $M = \max(|a|, |b|) \cdot c_{g'} \cdot \mathrm{Lip}(s) + \mathrm{Lip}(t)$. The Lipschitz constant of the inverse F^{-1} can be locally bounded for $y \in [a^*, b^*]^d$ as:

$$\mathrm{Lip}(F^{-1}) \leq \max(1, c_{\frac{1}{g}}) + M^*,$$

where $M^* = \max(|a^*|, |b^*|) \cdot c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) + c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) \cdot c_t + c_{\frac{1}{g}} \cdot \mathrm{Lip}(t)$.

Proof. We employ a similar proof strategy as in Lemma 5 and consider the Jacobian of the affine coupling layer. As in the proof of Lemma 5 we consider $I_1 = \{1, \ldots, m\}$ and $I_2 = \{m+1, \ldots, d\}$, where $1 < m < d$, without loss of generality. Since the structure of the forward and inverse mapping for affine coupling layers has some differences, we split the proof of the Lipschitz bounds into two steps. First, we start with the forward mapping and then reuse several steps for the bounds on the inverse mapping.

Derivation for the forward mapping:
The Jacobian of the forward affine block has the structure

$$J_F(x) = \begin{pmatrix} I_{m \times m} & 0_{(m) \times (d-m)} \\ D_I(x_{I_2}) D_{g'}(x_{I_1}) J_s(x_{I_1}) + J_t(x_{I_1}) & D_g(s(x_{I_1})) \end{pmatrix},$$

where D are the following diagonal matrices

$$\begin{aligned} D_I(x_{I_2}) &= \mathrm{diag}\left((x_{I_2})_1, \ldots, (x_{I_2})_{|I_2|}\right), \\ D_{g'}(x_{I_1}) &= \mathrm{diag}\left(g'(s(x_{I_2})_1, \ldots, g'(s(x_{I_2})_{|I_2|})\right), \\ D_g(s(x_{I_1})) &= \mathrm{diag}\left(g(s(x_{I_2})_1, \ldots, g(s(x_{I_2})_{|I_2|})\right), \end{aligned}$$

where $D_I(x_{I_2}), D_{g'}(x_{I_1}) \in \mathbb{R}^{(d-m) \times m}$ and $D_g(s(x_{I_1})) \in \mathbb{R}^{(d-m) \times (d-m)}$. To simplify notation, we introduce

$$M(x) = D_I(x_{I_2}) D_{g'}(x_{I_1}) J_s(x_{I_1}) + J_t(x_{I_1}).$$

By using an analogous derivation as in the proof of Lemma equation 6 (up to the inequality sign), we get:

$$\begin{aligned} \mathrm{Lip}(F)^2 &\leq \sup_{x \in [a,b]^d} \sup_{\|x^*\|_2 = 1} \|x^*_{I_1}\|_2^2 + \left(\|D_g(s(x_{I_1}))x^*_{I_2}\|_2 + \|M(x)x^*_{I_1}\|_2\right)^2 \\ &= \sup_{x \in [a,b]^d} \max_{i \in [|I_1|]} (1, D_g(s(x_{I_1})_i))^2 + 2 \max_{i \in [|I_1|]} (D_g(s(x_{I_1})_i)) \|M(x)\|_2 + \|M(x)\|_2^2 \\ &\leq \sup_{x \in [a,b]^d} \max_{i \in [|I_1|]} (1, D_g(s(x_{I_1})_i))^2 + 2 \max_{i \in [|I_1|]} (1, D_g(s(x_{I_1})_i)) \|M(x)\|_2 + \|M(x)\|_2^2 \\ &= \sup_{x \in [a,b]^d} \left(\max_{i \in [|I_1|]} (1, D_g(s(x_{I_1})_i)) + \|M(x)\|_2\right)^2 \\ \iff \mathrm{Lip}(F) &\leq \max_{i \in [|I_1|]} (1, D_g(s(x_{I_1})_i)) + \sup_{x \in [a,b]^d} \|M(x)\|_2. \end{aligned}$$

Next, we will look into the structure of $M(x)$ to derive a more precise bound. Since inputs x are assumed to be bounded as $x \in [a, b]^d$, it holds:

$$\|D_I(x_{I_2})\|_2 \leq \max(|a|, |b|).$$

Since we assumed that g is continuously differentiable, both g and g' will be bounded over closed intervals like $[a,b]^d$. We will denote these bounds as c_g and $c_{g'}$. Thus, for $x \in [a,b]^d$ it holds

$$\|D_{g'}(x_{I_1})\|_2 \leq c_{g'}.$$

In a similar manner as in Lemma 5, the spectral norm of the Jacobian of the scale-function s and translation-function t can be bounded by their Lipschitz constant, i.e.

$$\|J_s(x_{I_1})\|_2 \leq \mathrm{Lip}(s)$$
$$\|J_t(x_{I_1})\|_2 \leq \mathrm{Lip}(t).$$

By using the above bounds, we obtain

$$\sup_{x\in[a,b]^d} \|M(x)\|_2^2 \leq \max(|a|,|b|) \cdot c_{g'} \cdot \mathrm{Lip}(s) + \mathrm{Lip}(t),$$

which results in the local Lipschitz bounds for $x \in [a,b]^d$

$$\mathrm{Lip}(F) \leq \max(1, c_g) + \max(|a|,|b|) \cdot c_{g'} \cdot \mathrm{Lip}(s) + \mathrm{Lip}(t).$$

Derivation for the inverse mapping:
For the affine coupling block, the inverse is defined as:

$$F^{-1}(y)_{I_1} = y_{I_1}$$
$$F^{-1}(y)_{I_2} = (y_{I_2} - t(x_{I_1})) \oslash g(s(y_{I_1})),$$

where $g(\cdot) \neq 0$ for all X_{I_2}, I_1, I_2 as before and $\oslash$ denotes elementwise division. The Jacobian for this operation has the structure:

$$J_{F^{-1}}(x) = \begin{pmatrix} I & 0 \\ M^*(y) & D_{\frac{1}{g}}(s(x_{I_1})) \end{pmatrix},$$

where $D_{\frac{1}{g}}(s(x_{I_1}))$ denotes a diagonal matrix, as before. Furthermore, M^* is defined as:

$$M^*(y) = D_I(y_{I_2}) D_{\left(\frac{1}{g}\right)'}(s(y_{I_1})) J_s(y_{I_1}) - D_{\left(\frac{1}{g}\right)'}(s(y_{I_1})) J_s(y_{I_1}) D_I(t(y_{I_1})) - D_{\frac{1}{g}}(s(y_{I_1})) J_t(y_{I_1}),$$

where $D_{\left(\frac{1}{g}\right)'}(s(x_{I_1}))$ also denotes a diagonal matrix. Using analogous arguments as for the forward mapping, we obtain the bound:

$$\mathrm{Lip}(F^{-1}) \leq \max_{i\in[|I_1|]} (1, D_{\frac{1}{g}}(s(x_{I_1})_i)) + \sup_{y\in[a^*,b^*]^d} \|M^*(y)\|_2.$$

Hence, we need to further bound the spectral norm of M^*. Since we assumed that g is continuously differentiable, both $\frac{1}{g}$ and $\left(\frac{1}{g}\right)'$ will be bounded over closed intervals like $[a^*,b^*]^d$. Furthermore, translation t is assumed to be globally continuous and thus also bounded over closed intervals. We will denote these upper bounds by $c_{\frac{1}{g}}$, $c_{\left(\frac{1}{g}\right)'}$ and c_t. Then we obtain the bound:

$$\sup_{y\in[a^*,b^*]^d} \|M^*(y)\|_2^2 \leq \max(|a^*|,|b^*|) \cdot c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) + c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) \cdot c_t + c_{\frac{1}{g}} \cdot \mathrm{Lip}(t).$$

Hence, we can bound the Lipschitz constant of the inverse of an affine block over the interval $[a^*,b^*]^d$ as:

$$\mathrm{Lip}(F^{-1}) \leq \max_{i\in[|I_1|]} (1, c_{\frac{1}{g}}) + \max(|a^*|,|b^*|) \cdot c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) + c_{\left(\frac{1}{g}\right)'} \cdot \mathrm{Lip}(s) \cdot c_t + c_{\frac{1}{g}} \cdot \mathrm{Lip}(t).$$

□

B.1 Proof of Theorem 2

Proof. Proof of statement (i):
The larger bi-Lipschitz bounds of the affine models compared to the additive models follows directly from Lemmas 5 and 6, as the affine bounds have only additional non-zero components in their bounds.

Proof of statement (ii):
The global Lipschitz bound for additive models is given in Lemma 5. In Lemma 6 we also provide local bounds for $x \in [a, b]^d$ for affine models. What remains to be show, is that there are no bi-Lipschitz bounds that hold globally for $x \in \mathbb{R}^d$.

For this, consider a simplified affine model as $F(x_1, x_2) = x_1 f(x_2)$ with $\frac{dF}{dx_2} = x_1 \frac{df}{dx_2}$. This derivative is unbounded if x_1 is allowed to be arbitrarily large, hence there is no global bound.

The same argument caries over to the full affine model, since both forward and inverse Jacobian involve the terms $D_I(x_{I_2})$ and $D_I(y_{I_2})$, respectively (see proofs of Lemma 5 and 6). When x for the forward and y for the inverse are not assumed to be bounded, the Jacobian can have unbounded Frobenius norm. Not that this only holds if g and s are not constant functions, which is why we need to assume this property to hold (otherwise the affine model would collapse to an additive model). The unbounded Jacobian in turn induces a unbounded spectral norm due to the equivalence of norms in finite dimensions and thus no Lipschitz bound can be obtained. □

C DETAILS FOR 2D DENSITY MODELING EXPERIMENTS

Here we provide experimental details for the 2D checkerboard experiments from Section 4.1.1. The samples are shown in Figure 7, which shows that the data lies within $x_1 \in [-4, 4]$ and $x_2 \in [-4, 4]$ and exhibit jumps at the border of the checkerboard.

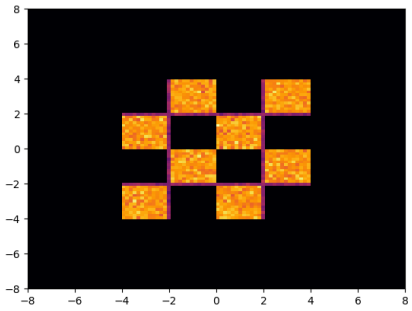

Figure 7: **Samples from 2D checkerboard.**

We train the following three large models using the residual flows repository and the corresponding hyperparameter settings[4]. For completeness, we provide the hyperparameters in Table 4 below.

Hyperparameter	Value
Batch Size	500
Learning Rate	1e-3
Weight Decay	1e-5
Optimizer	Adam
Hidden Dim	128-128-128-128
Num Blocks	100
Activation	swish
ActNorm	False

Table 4: **Hyperparameters for training 2D models on checkerboard data.**

To consider the effect of different architecture settings on stability we train three INN variants:

[4]from https://github.com/rtqichen/residual-flows/blob/master/train_toy.py

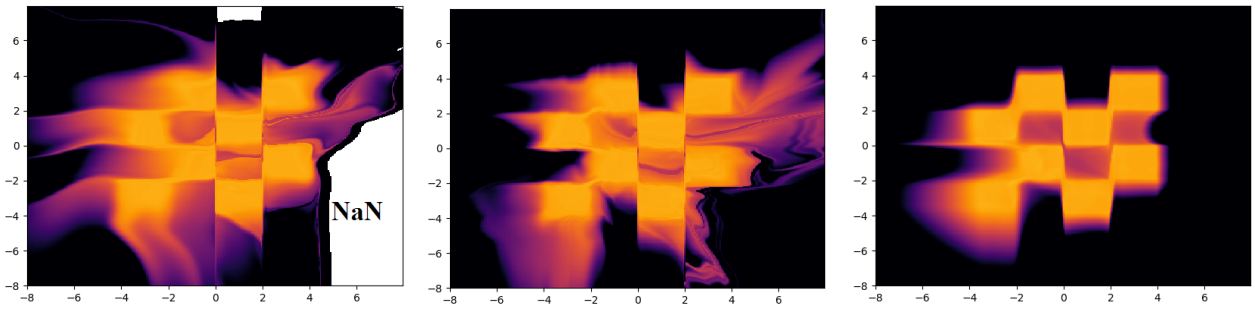

Figure 8: **Learned density on 2D checkerboard data: Left:** standard affine model with sigmoid scaling; **Middle**: more stable affine model with scaling in [0.5, 1]; **Right**: Residual Flow (Chen et al., 2019).

1. Affine coupling model with standard sigmoid scaling for the elementwise function g from equation 4, which results in a scaling in $(0, 1)$.

2. Modified affine coupling model with a scaling in $(0.5, 1)$ by a squashed sigmoid.

3. Residual flow (Chen et al., 2019) with a coefficient of 0.8 for spectral normalization to satisfy the contraction requirement from i-ResNets (Behrmann et al., 2019).

In addition to the reconstruction error in Figure 2 (main body of the paper), we visualize the learned density function for the models above in Figure 8. Most importantly, the instability a the affine model is clearly visible in the NaN density values outside the data domain. The more stable affine model with the modified scaling does not exhibit this failure, but still appears to learn a density with large slopes. The residual flow on the other hand learns a more stable distribution. Lastly, we note that the trained models on this 2D data were deliberately large to emphasize failures even in a low dimensional setting.

D DETAILS FOR OOD AND SAMPLE EVALUATION EXPERIMENTS

The examples from each OOD dataset were normalized such that pixel values fell into the same range as each model was originally trained on: $[-0.5, 0.5]$ for Glow, and $[0, 1]$ for Residual Flows. The OOD datasets were adopted from previous studies including those by Hendrycks and Gimpel (2016); Liang et al. (2017). Extended results are shown in Table 5. The pre-trained models we used for OOD evaluation were from the official Github repositories corresponding to the respective papers. The pre-trained Glow model used 1×1 convolutions.

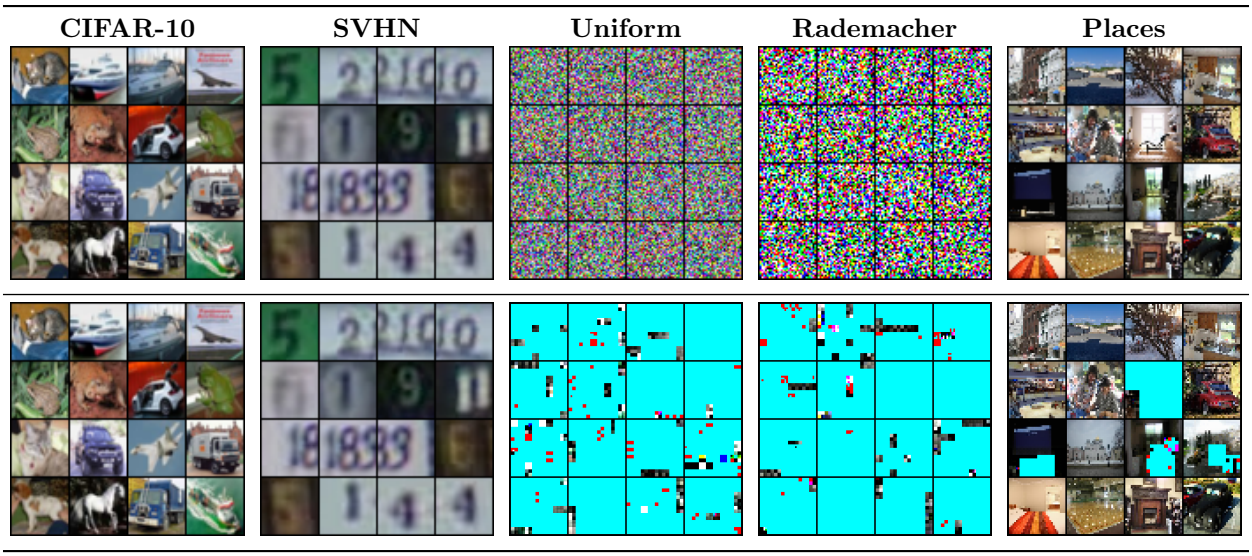

Table 5: **Reconstructions of OOD data, using a CIFAR-10 pre-trained Glow model.** Broken reconstructions that contain `inf` values are highlighted in cyan.

Details for Training Glow This section details the Glow models trained with the Flow/MLE objective reported in Section 4.1.2. These models have slightly different hyperparameters than the pre-trained Glow models. The controlled hyperparameters are listed in Table 6.

Hyperparameter	Value
Batch Size	64
Learning Rate	5e-4
Weight Decay	5e-5
Optimizer	Adamax
Flow Permutation	Reverse
# of Layers	3
# of Blocks per Layer	32
# of Conv Layers per Block	3
# of Channels per Conv	512

Table 6: **Hyperparameters for trained Glow.**

The LR is warmed up linearly for 5 epochs. The training data is augmented with 10% translation, and random horizontal flips. The additive and affine models used the coupling blocks described in Eq. 3 and Eq. 4.

Details for Evaluating Glow Stability statistics reported included reconstruction errors and condition numbers (Figure 9). Reconstruction error reported is the pixel-wise ℓ_2^2 distance measured in the $[-0.5, 0.5]$ range. Input and reconstruction pairs are visualized in Figure 10. Condition numbers are computed numerically as follows: 1. gradient w.r.t. each dimension of the network output is computed sequentially to form the Jacobian, 2. SVD of the Jacobian is computed using "numpy.linalg.svd", 3. the reported condition number is the ratio of the largest to smallest singular value.

E Non-Invertibility in Data Distribution

In this section we expose non-invertibility in the data distribution by optimizing within the dequantization distribution of a datapoint to find regions that are poorly reconstructed by the model. Note that the inputs found this way are valid samples from the training distribution. We performed this analysis with three NFs trained on CelebA64 with commonly-used 5-bit uniform dequantization: 1) affine Glow with standard sigmoid scaling; 2) affine Glow model with modified scaling in $(0.5, 1)$; and 3) a Residual Flow. Starting from an initial training

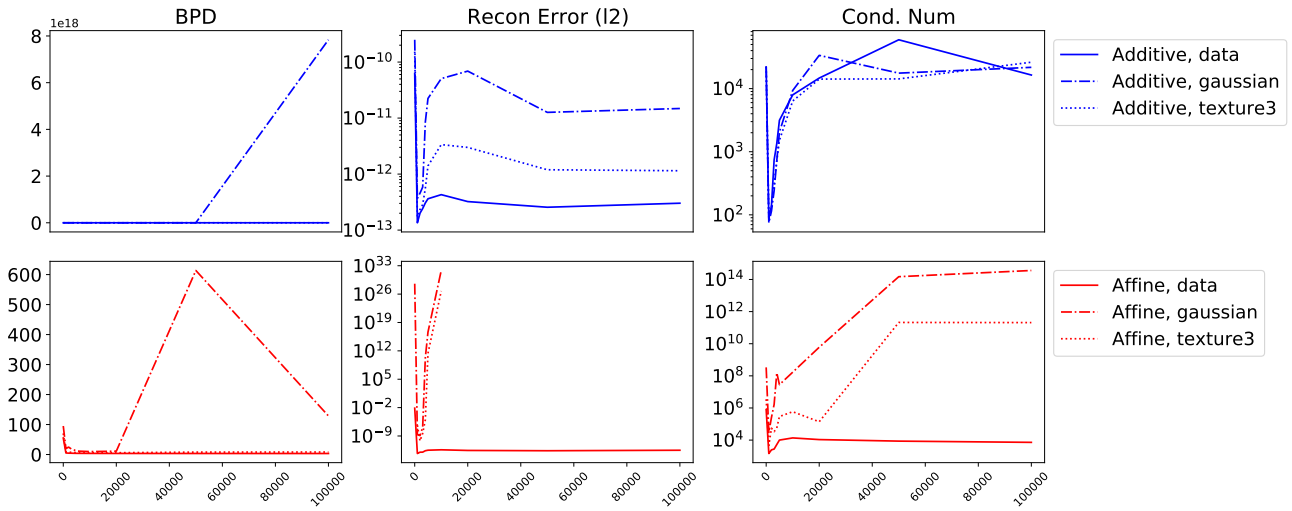

Figure 9: **Stability statistics of Flow model** over gradient steps on the x-axis.

datapoint x, we optimized in input-space to find a perturbed example x' that induces large reconstruction error using Projected Gradient Descent (Madry et al., 2018):

$$\underset{||x'-x||_\infty<\epsilon}{\arg\max} \; ||x' - F^{-1}(F(x'))||_2, \tag{11}$$

where ϵ is determined by the amount of uniform dequantization, see Appendix E.1 for details. As shown in Figure 11, this attack reveals the instability of affine models and underlines the stability of Residual Flows (Chen et al., 2019). To conclude, we recommend Residual Flows for a principled application of NFs.

E.1 Experimental Details of Invertibility Attack

In order to probe the invertibility of trained flow models over the entire data distribution, we trained three INN models on CelebA64 (with 5-bit dequantization) using the residual flows repository and the corresponding hyperparameter settings[5]. For completeness, we provide the hyperparameters in the Table 7.

Hyperparameter	Value
Batch Size	64
Learning Rate	1e-3
Weight Decay	0
Optimizer	Adam (ResFlow), Adamax (Affine)
Warmup iter	1000
Inner Dim	512
Num Blocks	16-16-16-16
ActNorm	True
Activation	Swish (ResFlow), ELU (Affine)
Squeeze First	True
Factor Out	False
FC end	True
Num Exact Terms	8
Num Trace Samples	1
Padding Distribution	uniform

Table 7: **Hyperparameters for training affine and ResFlow models on CelebA64.**

[5]from https://github.com/rtqichen/residual-flows/blob/master/train_img.py

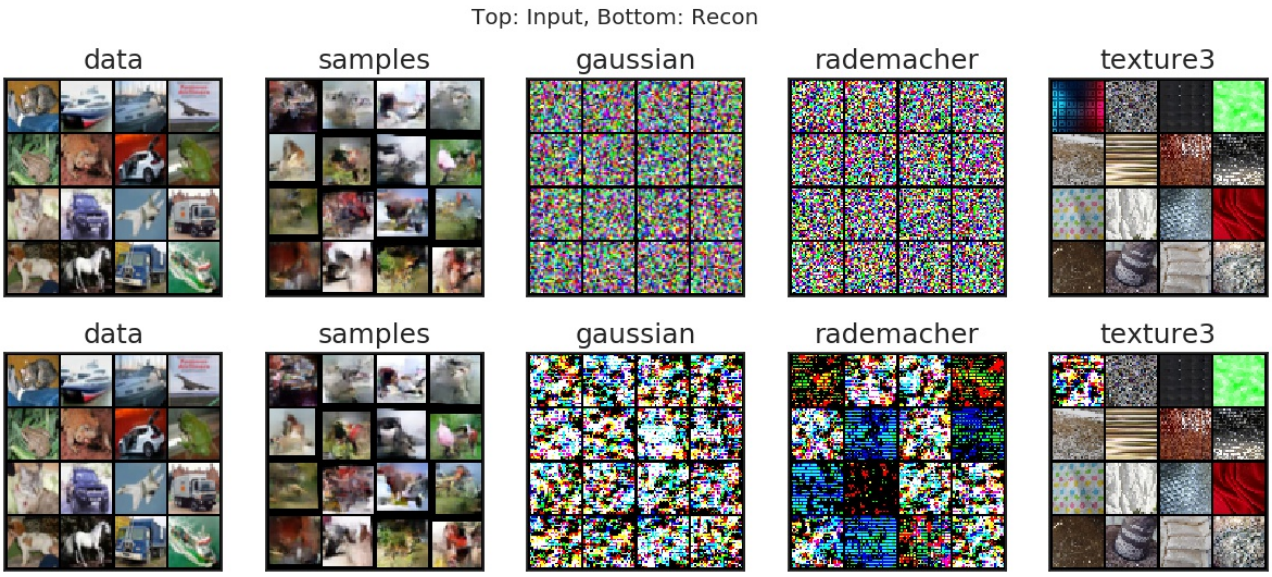

Figure 10: **Reconstructions using Affine model.** Top row is the input, and bottom row is the reconstruction. In the case of perfect reconstruction, the two rows should look identical. Notice that the model is not only non-invertible for the white noise, the top-left image of "texture 3" also fails to be reconstructed. Some of the images in "samples" also look slightly different.

For evaluation we used the checkpoint obtained via tracking the lowest test bits-per-dimension. As a summary, we obtained the following three models:

1. An affine coupling model with standard sigmoid scaling for the elementwise function g from Eq. 4, which results in a scaling in $(0, 1)$.
2. A modified affine coupling model with a scaling in $(0.5, 1)$ by a squashed sigmoid.
3. A Residual Flow (Chen et al., 2019) with a coefficient of 0.98 for spectral normalization to satisfy the contraction requirement from i-ResNets (Behrmann et al., 2019).

To explore the data distribution, we first need to consider the pre-processing step that is used to turn the quantized digital images into a continuous probability density: dequantization, see e.g. (Ho et al., 2019) for a recent discussion on dequantization in normalizing flows. Here we used the common uniform dequantization with 5-bit CelebA64 data, which creates noisy samples:

$$\hat{x} = \frac{x + \delta}{2^5}, \text{ where } \quad x \in \{0, \dots, 2^5\}^d \quad \text{and} \quad \delta \sim \text{uniform}\{0, 1\},$$

where $d = 64 \cdot 64 \cdot 3$. Thus, around each training sample x there is a uniform distribution which has equal likelihood under the data model. If we explore this distribution with the invertibility attack below, we can guarantee by design that we stay within the data distribution. This is fundamentally different to the classical setting of adversarial attacks (Szegedy et al., 2013), where it is unknown if the crafted inputs stay within the data distribution.

These models were then probed for invertibility with the data distribution using the following attack objective:

$$\underset{||x' - \hat{x}||_\infty < \epsilon}{\arg\max} \; L(x') := ||x' - F^{-1}(F(x'))||_2, \tag{12}$$

where $\epsilon = \frac{0.5}{2^5}$, $\hat{x} = \frac{x+0.5}{2^5}$ and x was selected from the training set of CelebA64. By using this ℓ_∞ constraint, we thus make sure that we explore only within the data distribution. The optimization was performed using projections to fulfill the constraint and by signed gradient updates as:

$$x'_{k+1} = x'_k + \alpha \, \text{sign}(\nabla_{x'} L(x'_k)),$$

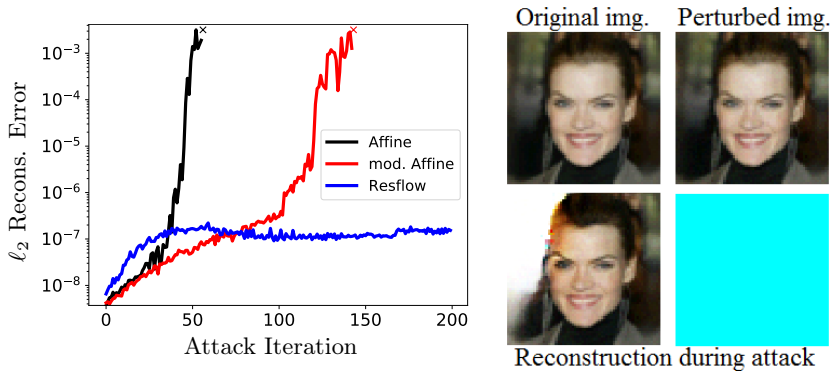

Figure 11: **Non-invertibility within the data-distribution of CelebA64 revealed by an invertibility attack. Left:** ℓ_2-reconstruction error obtained by the attack. The reconstructions of both affine models explode to `NaN` (indicated by $\times$), while the Residual Flow remains stable. **Right:** Despite no visual differences between the original and perturbed image, reconstruction fails for the affine model.

Figure 12: **Checking invertibility of Residual Flow on CelebA64.** From left to right, the images show: original image, perturbed image within dequantization range, reconstruction with 5 fixed point iterations, reconstruction with converged fixed point iterations. While the attack is able to find regions in the data distribution, which exhibit reconstruction error, the error is only due to a limited number of inversion steps.

with $\alpha = 5e^{-4}$ and $k = \{1, \dots, 200\}$.

In addition to the visual results in Figure 11, we present visual results from the Residual Flow model in Figure 12. For this model, we have to remark one important aspect: due to memory constraint on the used hardware (NVIDIA GeForce GTX 1080, 12 GB memory), we could only use 5 fixed-point iterations for the inverse of the i-ResNet (Behrmann et al., 2019), since computing backprop through this iteration is memory intensive. This is why, we observe visible reconstruction errors in Figure 12, which are due to a small number of iterations as the reconstruction on the right with more iterations shows. However, we note that the comparison is not entirely fair since the gradient-based attack has only access to the 5-iteration inverse, while for affine models it has access to the full analytical inverse. Thus, future work should address this issue e.g. by considering non-gradient based attacks.

Furthermore, we note that the bi-Lipschitz bounds for i-ResNets from (Behrmann et al., 2019) (see also overview Table 3) guarantee stability, which is why using more sophisticated attacks for the Residual Flow model should not result in invertibility failures as in affine models.

F EXPERIMENTAL DETAILS FOR 2D TOY REGRESSION

In this section we provide experimental details for the 2D toy regression experiment in Section 4.2.

Data. To generate the toy data, we sampled $10,000$ input-target pairs $((x_1, y_1), (x_2, y_2))$ distributed according to the following multivariate normal distributions:

$$(x_1, y_1) \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix}\right) \tag{13}$$

$$(x_2, y_2) \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\right). \tag{14}$$

We used ϵ =1e-24 so that we are essentially mapping $\mathbf{x}$ onto a 1D subspace.

Model. We used an affine Glow model with reverse permutations and ActNorm (Kingma and Dhariwal, 2018), where each block was a multi-layered perceptron (MLP) with two hidden layers of 128 units each and ReLU activations. We trained the model in full-batch mode using Adam with fixed learning rate 1e-4 for 40,000 iterations.

Additional Results. Figure 13 compares the mean squared error (MSE) loss, numerical reconstruction error, and condition number of the unregularized and regularized Glow models trained on this toy task. The regularized model adds the normalizing flow objective with a small coefficient of 1e-8. Here we see that the regularized model still achieves low MSE, while remaining stable with reconstruction error more than four orders of magnitude smaller than the unregularized model, and a condition number six orders of magnitude smaller.

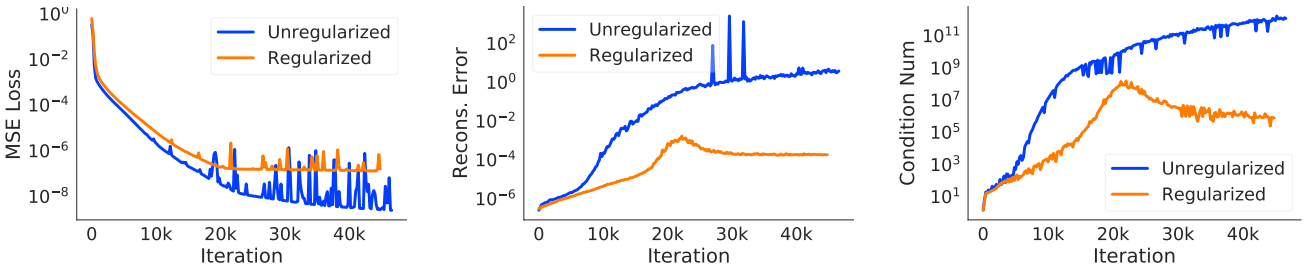

Figure 13: **A comparison of the mean-squared error (MSE) loss, reconstruction error, and condition number of an unregularized and a regularized Glow model trained on the toy 2D regression task.** The regularized model uses the normalizing flow objective with coefficient 1e-8.

G EXPERIMENTAL DETAILS FOR CLASSIFICATION EXPERIMENTS

In this section we provide details on the experimental setup for the classification experiments in Section 4.2, as well as additional results. We used the PyTorch framework (Paszke et al., 2019) for our INN implementations. All experiments were run on NVIDIA Titan Xp GPUs.

Experimental Setup. All the additive and affine coupling-based models we used have the same architecture, that consists of 3 levels, 16 blocks per level, and 128 hidden channels. Each level consists of a sequence of residual blocks that operate on the same dimensionality. Between levels, the input is spatially downsampled by 2× in both width and height, while the number of channels is increased by 4×. Each block consists of a chain of $3 \times 3 \rightarrow$ ReLU $\rightarrow 1 \times 1 \rightarrow$ ReLU $\rightarrow 3 \times 3$ convolutions. Because the dimension of the output of an INN is equal to that of the input (e.g., we have a 3072-dimensional feature space for $3 \times 32 \times 32$ CIFAR-10 images), we use a projection layer (1D BatchNorm $\rightarrow$ ReLU $\rightarrow$ Linear) to map the feature representation to 10 dimensions representing class logits. We trained the models on CIFAR-10 for 200 epochs, using Adam with initial learning rate 1e-4, decayed by a factor of 10 at epochs 60, 120, and 160 (following (Zhang et al., 2019)). We used standard data normalization (transforming the data to have zero mean and unit variance), and data augmentation (random cropping and horizontal flipping). The hyperparameters we used are summarized in Table 8.

Regularization Hyperparameters. We performed grid searches to find suitable coefficients for the regularization schemes we propose. In particular, we searched over coefficients { 1e-3, 1e-4, 1e-5 } for the normalizing flow regularizer, and { 1e-3, 1e-4, 5e-5 } for bi-directional finite-differences (FD) regularization. The effects of different regularization strengths are summarized in Table 9. We also plot the condition numbers, maximum/minimum singular values, and test accuracies while training with different coefficients for the normalizing flow objective (Figure 14) and bi-directional FD (Figure 15). For both regularization methods, larger coefficients (e.g., stronger regularization) yield more stable models with smaller condition numbers, but too large a coefficient can harm performance (i.e., by hindering model flexibility). For example, using a large coefficient (1e-3) for the normalizing flow objective substantially degrades test accuracy to 83.28%, compared to 89.07% for the un-regularized model.

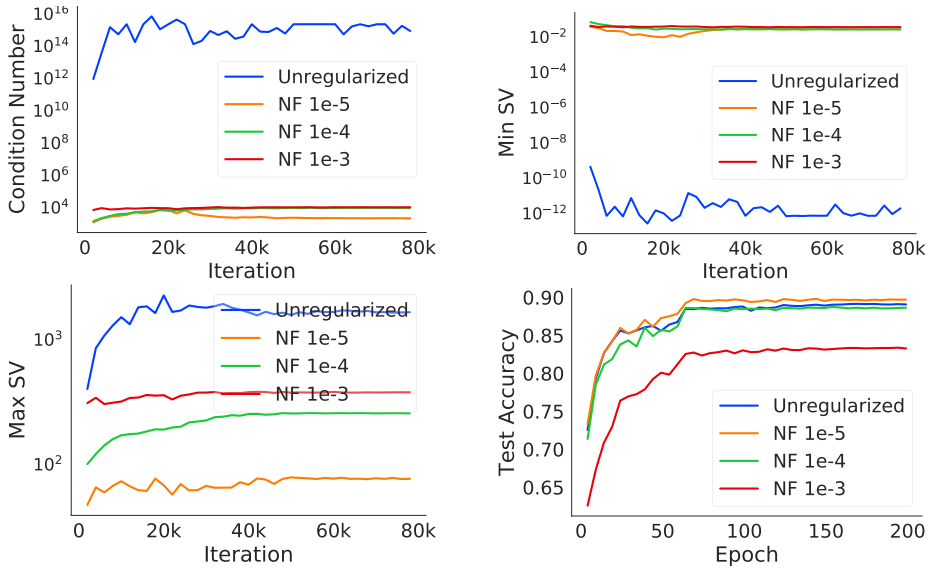

Figure 14: Comparison of regularization strengths for the **normalizing flow objective** during training of a Glow-like architecture (affine coupling, with 1×1 convolutions and ActNorm) on CIFAR-10. **Top-Left:** condition numbers for the un-regularized and regularized models; **Top-Right/Bottom-Left:** min/max singular values of the Jacobian; **Bottom-Right:** test accuracies.

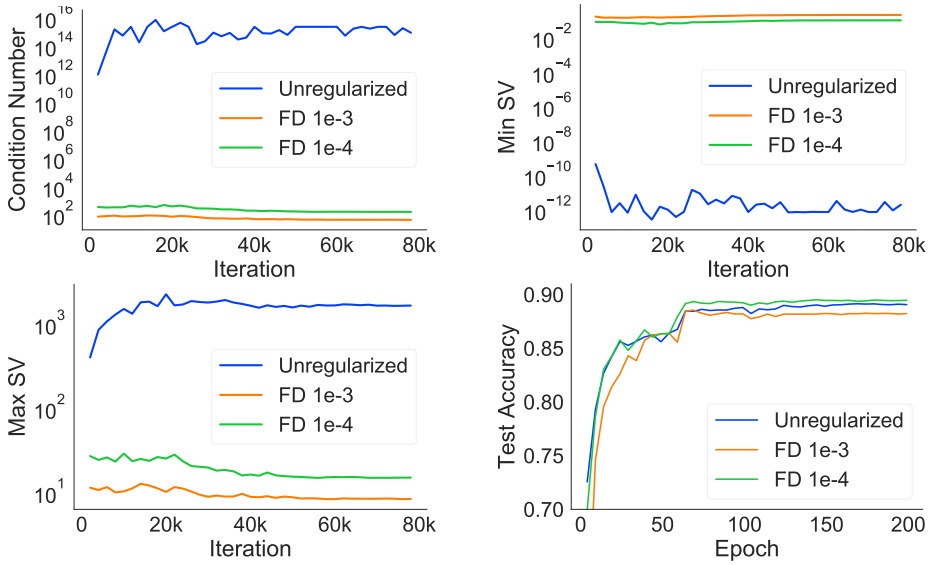

Figure 15: Comparison of regularization strengths for the **bi-directional finite differences (FD) regularizer** during training of a Glow-like architecture with affine coupling and 1×1 convolutions on CIFAR-10. **Top-Left:** condition numbers for the un-regularized and regularized models; **Top-Right/Bottom-Left:** min/max singular values of the Jacobian; **Bottom-Right:** test accuracies.

Hyperparameter	Value
Batch Size	128
Learning Rate	1e-4 (decayed by 10x at epochs $\{60, 120, 160\}$)
Weight Decay	0
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$)
# of Layers	3
# of Blocks per Layer	16
# of Conv Layers per Block	3
# of Channels per Conv	128

Table 8: **Hyperparameters for INN classifiers.**

Extended Results. We provide an extended version of Table 2 (from the main paper) in Table 10, where we also include additive and affine models with shuffle permutations. Figures 16 and 17 visualize the stability during training using each regularizer, for additive and affine models, respectively.

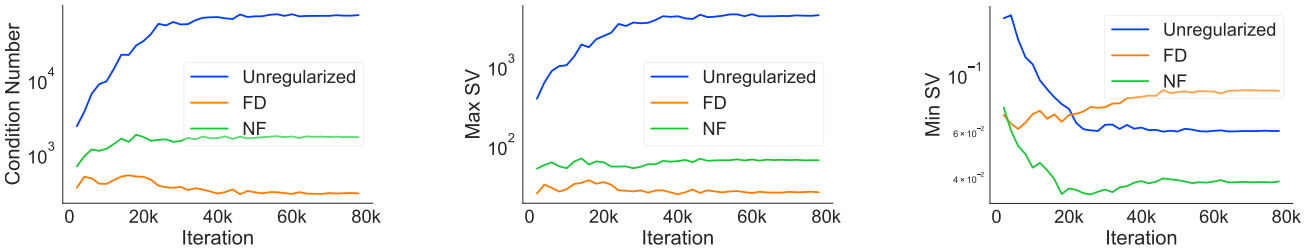

Figure 16: Stability during training of a Glow-like architecture with **additive coupling** and 1×1 **convolutions** on CIFAR-10. **Left:** condition numbers for the un-regularized and regularized models. **Middle/Right:** min/max singular values of the Jacobian. Note the large effect of both regularizes on the min singular value, indicating a more stable inverse mapping.

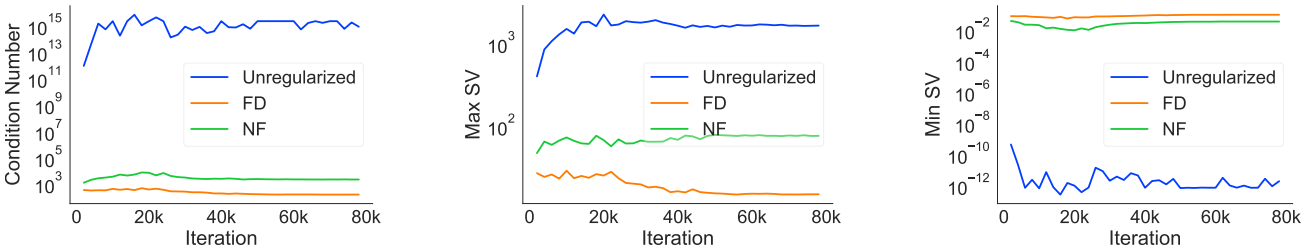

Figure 17: Stability during training of a Glow-like architecture with **affine coupling** and 1×1 **convolutions** on CIFAR-10. **Left:** condition numbers for the un-regularized and regularized models. **Middle/Right:** min/max singular values of the Jacobian.

Method	Coeff.	Inv?	Test Acc	Recons. Err.	Cond. Num.	Min SV	Max SV
Add. C, None	0	✓	89.73	4.3e-2	7.2e+4	6.1e-2	4.4e+3
Add. C, FD	1e-3	✓	88.34	5.9e-4	4.3e+1	2.1e-1	9.1e+0
Add. C, FD	1e-4	✓	89.10	9.3e-4	2.0e+2	9.8e-2	2.0e+1
Add. C, FD	5e-5	✓	89.71	1.1e-3	3.0e+2	8.7e-2	2.6e+1
Add. C, NF	1e-3	✓	84.11	5.5e-4	6.5e+3	3.2e-2	2.1e+2
Add. C, NF	1e-4	✓	88.65	6.5e-4	4.7e+3	3.4e-2	1.6e+2
Add. C, NF	1e-5	✓	89.52	9.9e-4	1.7e+3	3.9e-2	6.6e+1
Aff. S, None	0	✗	87.64	`NaN`	9.9e+13	1.7e-12	1.6e+2
Aff. S, FD	1e-3	✓	86.85	2.0e-5	4.0e+1	1.8e-1	7.1e+0
Aff. S, FD	1e-4	✓	88.14	3.1e-5	1.2e+2	1.1e-1	1.3e+1
Aff. S, FD	5e-5	✓	88.13	3.9e-5	1.8e+2	9.6e-2	1.7e+1
Aff. S, NF	1e-3	✓	80.75	3.1e-5	2.2e+4	2.0e-2	4.4e+2
Aff. S, NF	1e-4	✓	87.87	2.3e-5	5.8e+3	3.1e-2	1.8e+2
Aff. S, NF	1e-5	✓	88.31	2.8e-5	8.5e+2	3.7e-2	3.2e+1
Aff. C, None	0	✗	89.07	`Inf`	8.6e+14	1.9e-12	1.7e+3
Aff. C, FD	1e-3	✓	88.24	6.0e-4	4.2e+1	2.0e-1	8.4e+0
Aff. C, FD	1e-4	✓	89.47	9.6e-4	1.6e+2	9.6e-2	1.5e+1
Aff. C, NF	1e-3	✓	83.28	7.0e-4	1.0e+4	3.6e-2	3.8e+2
Aff. C, NF	1e-4	✓	88.64	8.6e-4	9.6e+3	2.7e-2	2.6e+2
Aff. C, NF	1e-5	✓	89.71	1.3e-3	2.2e+3	3.5e-2	7.7e+1

Table 9: **Effect of the regularization coefficients for both finite differences regularization (denoted FD) and the normalizing flow regularizer (NF), when training several INN classifiers on CIFAR-10.** All experiments in this table used Glow-like architectures with either additive or affine coupling, and either shuffle permutations or 1×1 convolutions. In the Method column, "Add." and "Aff." denote additive and affine coupling, respectively; "C." and "S." denote 1×1 convolutions and shuffle permutations, respectively. Note that for the affine model with 1×1 convolutions, FD coefficient 5e-5 was too small to ensure stabilization, so it is not included above.

Finite Differences Regularization. As mentioned in Section 3.2, for the bi-directional finite differences regularizer we used samples $v \sim \mathcal{N}(0, I)$. For the step size in Eq. 6, we used $\epsilon = 0.1$ to avoid numerical errors due to *catastrophic cancellation*.

Computational Efficiency of Regularizers. Most invertible neural networks are designed to make it easy to compute the log determinant of the Jacobian, needed for the change-of-variables formula used to train flow-based generative models. For coupling-based INNs, the log determinant is particularly cheap to compute, and can be done in the same forward pass used to map $x \mapsto z$. Thus, adding a regularizer consisting of the weighted normalizing flow loss does not incur any computational overhead compared to standard training.

Bi-directional finite differences regularization is more expensive: memory-efficient gradient computation involves a forward pass, an inverse pass, and a backward pass. The forward regularizer adds an additional overhead to these computations because it requires the previous computations not only for clean x, but also for noisy $x + \epsilon v$. The inverse regularizer also passes two variables $z = F(x)$ and $\hat{z} = F(x) + \epsilon v^*$ through its computations, which are: an inverse pass to compute the reconstruction, a forward pass to re-compute activations of the inverse mapping and the backward pass through the inverse and forward. However, in practice the cost can be substantially reduced by applying regularization only once per every K iterations of training; we found that $K = 5$ and $K = 10$ performed similarly to $K = 1$ in terms of their stabilizing effect, while requiring only a fraction more computation (e.g., 1.26× the computation when $K = 10$) than standard training. See the wall-clock time comparison in Table 11.

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.

Model	Regularizer	Inv?	Test Acc	Recons. Err.	Cond. Num.	Min SV	Max SV
Additive Shuffle	None	✓	88.35	1.1e-4	2.1e+3	2.9e-2	6.0e+1
	FD	✓	88.49	5.4e-5	7.0e+2	5.3e-2	3.7e+1
	NF	✓	88.49	2.2e-5	1.1e+3	3.0e-2	3.3e+1
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 Shuffle	None	✗	87.64	NaN	9.9e+13	1.7e-12	1.6e+2
	FD	✓	88.14	3.1e-5	1.2e+2	1.1e-1	1.3e+1
	NF	✓	88.31	2.8e-5	8.5e+2	3.7e-2	3.2e+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 10: **Extended results: Effect of regularization when training several additive and affine INN architectures for CIFAR-10 classification.**

Method	Time per Epoch (s)	Overhead
Unregularized	281.0	1×
Reg. Every 1	854.1	3.04×
Reg. Every 5	406.6	1.45×
Reg. Every 10	354.6	1.26×

Table 11: **Timing comparison for applying bi-directional finite differences regularization at different frequencies during training (every 1, 5, or 10 iterations).** Time is measured in seconds and corresponds to a single training epoch. All rows use memory-saving gradient computation; we used additive models with 1×1 convolutions, as these are also trainable without regularization, allowing us to time unregularized memory-saving gradients.

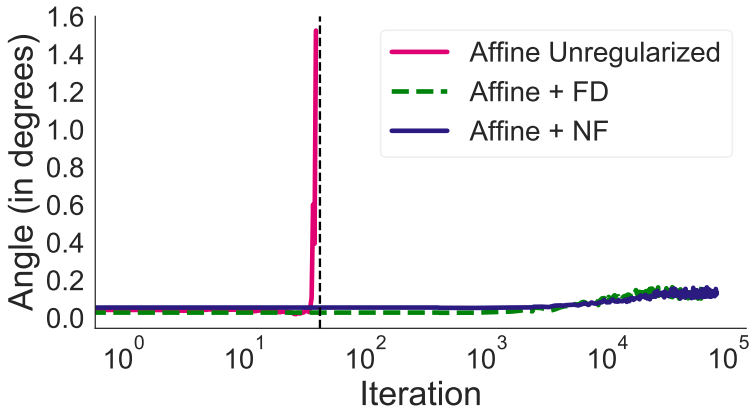

Figure 18: **Comparison of the angle between memory-saving and true gradients during training of a Glow model with affine coupling and 1×1 convolutions on CIFAR-10.** While the unregularized affine model suffers from an exploding inverse (the angle rapidly increases at the start of training and becomes NaN after the dashed vertical line), adding either finite differences (FD) or normalizing flow (NF) regularization stabilizes the model and keeps the angles small throughout training.

Comparing True and Memory-Saving Gradients. For the gradient angle figure in the introduction (Section 1), we used additive and affine Glow models with 1×1 convolutions. To create that figure, we first trained both models with exact (non-memory-saving) gradients, and saved model checkpoints during training; then, we loaded each checkpoint and computed the angle between the true and memory-saving gradients for a fixed set of 40 training mini-batches and plotted the mean angle. We did not train those models with memory-saving

gradients, because this would have made it impossible to train the unstable affine model (due to the inaccurate gradient).

Figure 18 shows a similar analysis, comparing the angles between true and memory-saving gradients for an affine model trained with and without regularization. The regularized models were trained with memory-saving gradients, while the unregularized model was trained with standard backprop. Similarly to the introduction figure, we saved model checkpoints throughout training, and subsequently loaded the checkpoints to measure the angle between the angles for the 40 training mini-batches, plotting the mean angle. Applying either finite differences or normalizing flow regularization when training the affine model keeps the angles small, meaning that the memory-saving gradient is accurate, and allowing us to train with this gradient.

H OUTLOOK ON BI-DIRECTIONAL TRAINING WITH FLOW-GAN

Experimental Details. Table 12 summarizes the hyperparameters for all models trained in our Flow-GAN experiments. The INN architecture in this experiment is similar to those described in Appendix D. Table 12 lists hyperparameters changed for this experiment.

Hyperparameter	Value
Batch Size	32
Learning Rate (Generator)	5e-5
Learning Rate (Discriminator)	5e-5
Weight Decay (Both)	0
Optimizer	Adam($\beta_1 = 0.5$, $\beta_2 = 0.99$)
# of Layers	3
# of Blocks per Layer	8
# of Conv Layers per Block	3
# of Channels per Conv	128

Table 12: **Hyperparameters for ADV models**

The prior is a standard normal. One extra hyperparameter here is the weight of the MLE loss (which is considered as a regularizer here). The discriminator had 7 convolution layers, each is regularized by Spectral normalization (Miyato et al., 2018) and using the LeakyReLU activation. The depth of the convolution layers are (64, 64, 128, 128, 256, 256, 512). The discriminator is further regularized with gradient penalty (Gulrajani et al., 2017) and optimized using the binary cross entropy GAN loss.

Stability at Data. Similarly to the results in the classification Section 4.2, when a flow is trained as a GAN, it suffers from non-invertibility even at training points. This is easily explained as the INN generator is never explicitly trained at the data points, but only receives gradients from the discriminator.

Using the MLE objective on the generator alleviates this issue. It increases the stability at the data-points, and makes training more stable in general, as discussed in section 3.3.2. This is exactly the FlowGAN formulation proposed in (Grover et al., 2018). See Table 13 for a comparison of test-set likelihoods and sample quality. In contrast to (Grover et al., 2018), we find additive Glow models trained with the GAN objective alone to be very unstable. We used a weighing of 1e-3 for the MLE loss for the reported results here. One suspicion is that the introduced ActNorm in the Glow architecture is causing extra instability.

Originally, one hypothesis was that the astronomically large BPD when trained as a GAN was caused by instability. This is likely incorrect. Even for stable runs (with a smaller, more stable architecture) that had no reconstruction errors and small condition numbers, the model still assigns incredibly large BPDs.

Our FlowGAN models strike a good balance between good test-set likelihood and improving sample quality. See Figure 19 for samples. The FlowGAN samples show better "objectness" and do not have the high-frequency artifacts shown in the MLE samples.

Unfortunately, given the fact that the GAN-only objective assigns unexplained BPDs, it's still unclear whether interpreting the BPDs for the FlowGAN in the same way as MLE models is recommended.

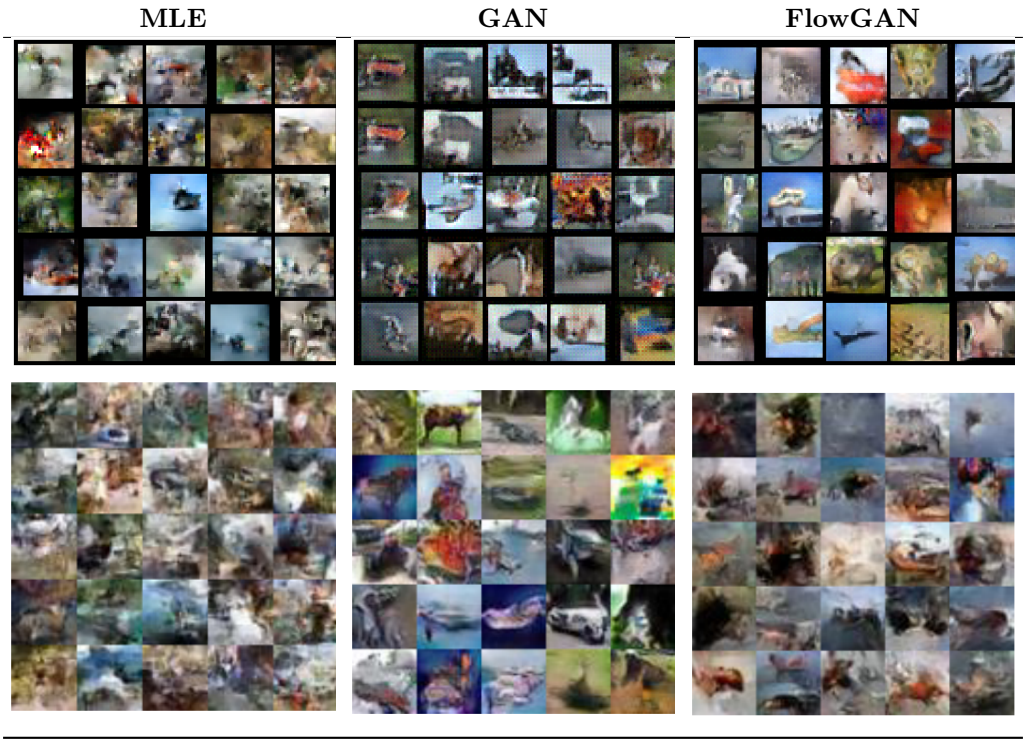

Figure 19: **Comparing samples from our FlowGANs to figures copied from (Grover et al., 2018) (bottom row).**

Objective	Inception Score	BPD
MLE	2.92	3.54
GAN	5.76	8.53
FlowGAN	3.90	4.21
-	**FID**	**BPD**
MLE	790	3.77
GAN*	850	Inf
FlowGAN	420	4

Table 13: **Comparing test-set likelihood and sample quality.** Top half of table copied from (Grover et al., 2018). *We find using only the GAN objective to be very unstable. Training can diverge after recording reasonable FID. One definitive difference between our results and (Grover et al., 2018) is BPD using only the GAN objective. Our runs often get infinite BPDs in a few hundred iterations, whereas (Grover et al., 2018) reports a BPD of 8.53. After inspecting their Table 2 together with Figure 2, the BPD and Inception Score seem to be reported for different times during training.