Understanding and Mitigating Exploding Inverses in Invertible Neural Networks

Jens Behrmann*, Paul Vicol*, Kuan-Chieh Wang*, Roger Grosse, Jorn Jacobsen

Slides by: Paul Vicol

Universität Bremen

UNIVERSITY OF TORONTO

VECTOR INSTITUTE

Outline

- Motivation
- Lipschitz Properties of INN Building Blocks
- Controlling Global Stability
- Controlling Local Stability
  - Bi-directional finite differences regularization
  - Normalizing Flow Regularization
- Experiments
  - Instability on OOD Data
  - Non-invertibility within the dequantization region
  - Memory-efficient gradient computation
- Summary and Practical Takeaways

Outline

- Motivation
- Lipschitz Properties of INN Building Blocks
- Controlling Global and Local Stability
  - Bi-directional finite differences regularization
  - Normalizing Flow Regularization
- Experiments
  - Instability on OOD Data
  - Non-invertibility within the dequantization region
  - Memory-efficient gradient computation
- Summary and Practical Takeaways

Applications of INNs

- The *application space for invertible neural networks (INNs) is growing rapidly*
  - Training generative models with exact likelihoods --- normalizing flows
  - Computing memory-saving gradients
  - Increasing posterior flexibility in VAEs
  - Solving inverse problems
  - Analyzing adversarial robustness

- However, as practitioners apply *off-the-shelf INNs to new problems w/ new objectives*, they often run into *stability issues that break the models*
  - Even worse, *many of these failures are not immediately apparent during training*

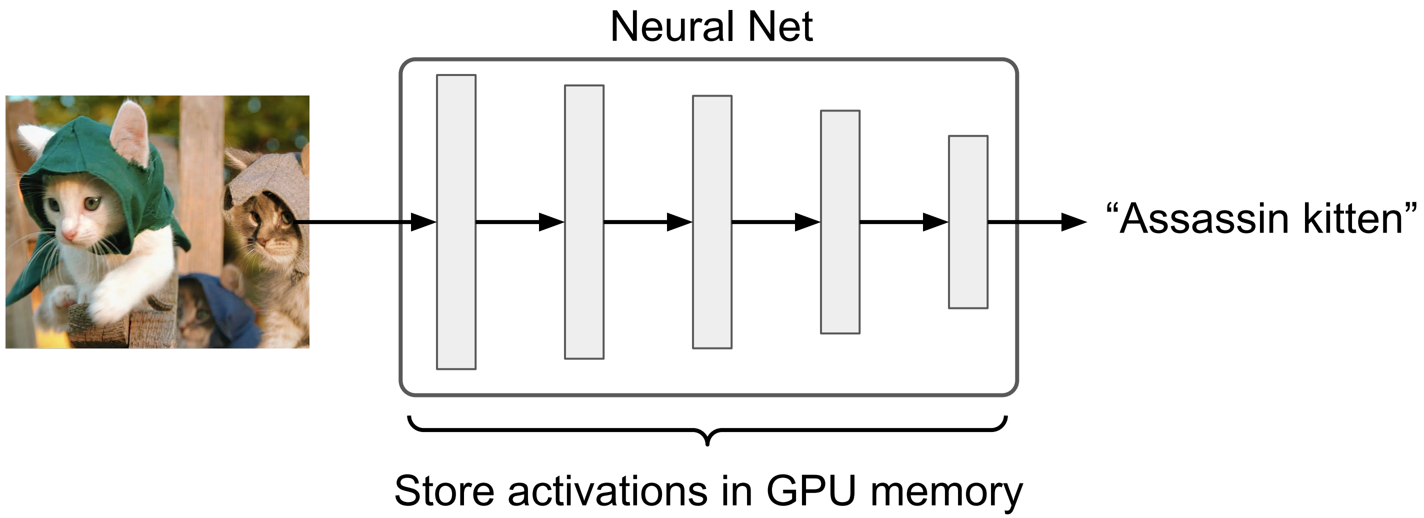
Applications of INNs

- The *application space for invertible neural networks (INNs) is growing rapidly*

Our Focus
- Training generative models with exact likelihoods --- normalizing flows
- Computing memory-saving gradients
- Increasing posterior flexibility in VAEs
- Solving inverse problems
- Analyzing adversarial robustness

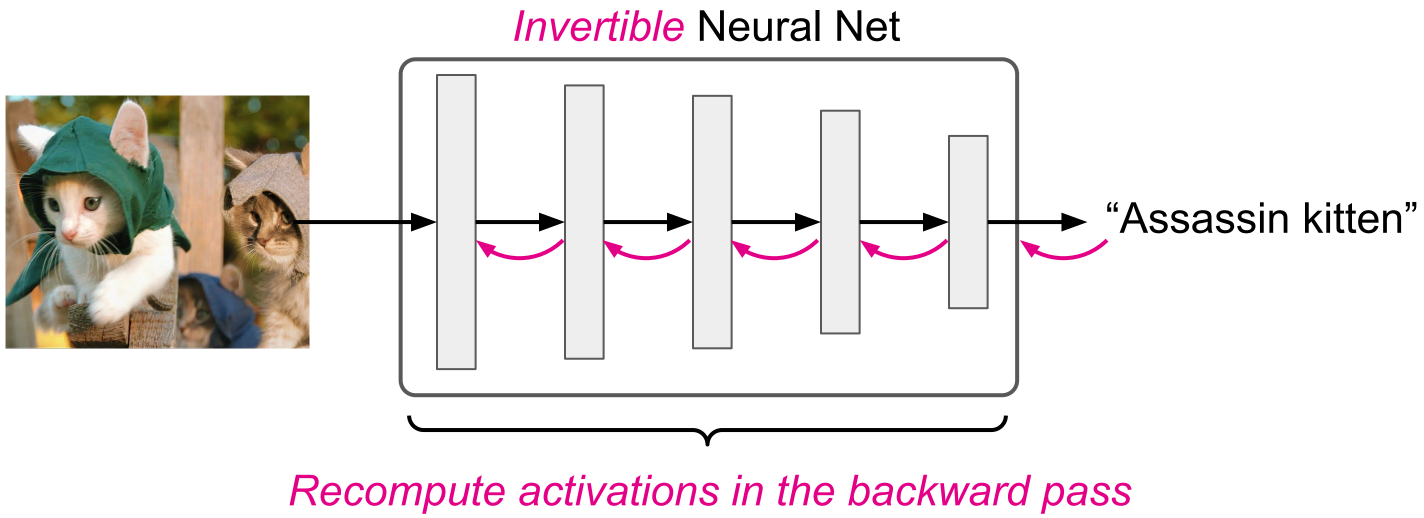
- However, as practitioners apply *off-the-shelf INNs to new problems w/ new objectives*, they often run into *stability issues that break the models*
    - Even worse, *many of these failures are not immediately apparent during training*

Memory-Efficient Gradient Computation



- Typically, we *store the intermediate activations of a neural net in memory* to compute gradients in the backward pass
- Activation memory is often a limiting factor when using:
    1. Large images (e.g., medical images)
    2. Large minibatches
    3. Deep models

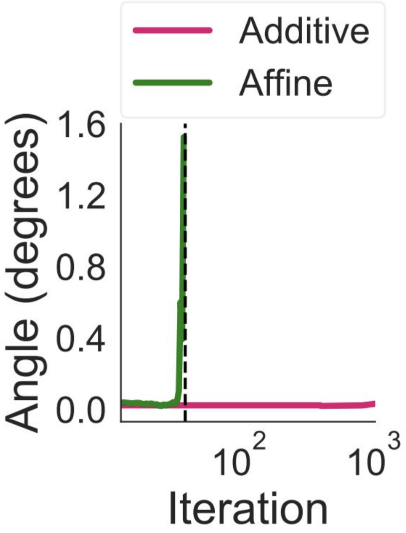
Memory-Efficient Gradient Computation



- With an INN, you don’t need to store intermediate activations in memory
    - You can *reconstruct activations during the backward pass*, trading off reduced memory for increased computation
- **Key assumption:** the INN is numerically stable, so that the reconstructed activations are equivalent to the ones from the forward pass

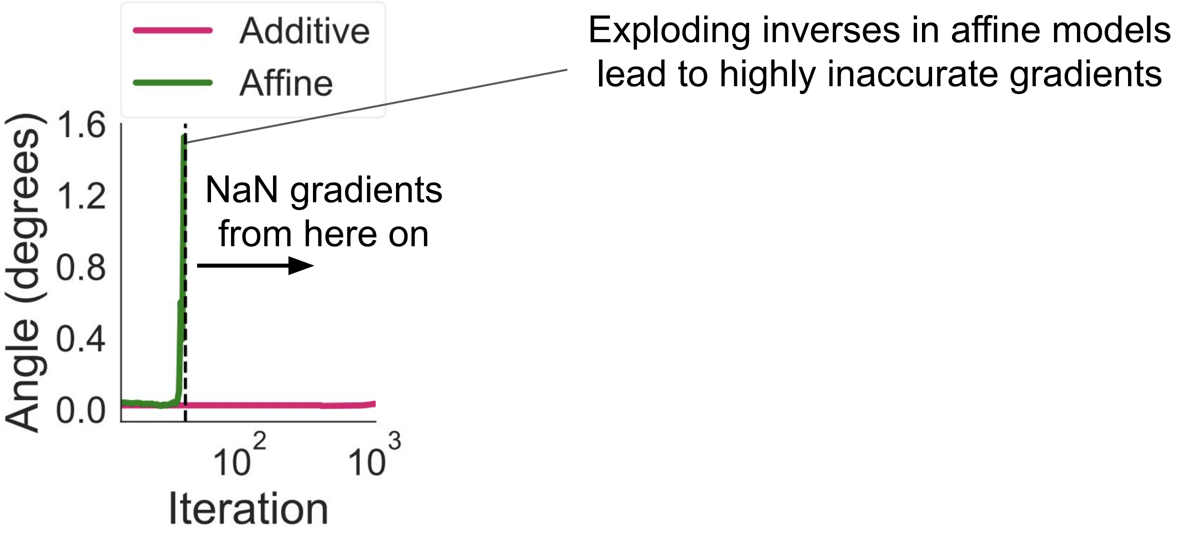
Motivation: Issues with Memory-Saving Gradients

- We can save memory by discarding activations, and *recomputing them in the backward pass, e.g., "memory-saving gradients"*
- Measure the quality of the memory-saving gradient by computing the *angle to the true gradient* (that is computed using stored activations)



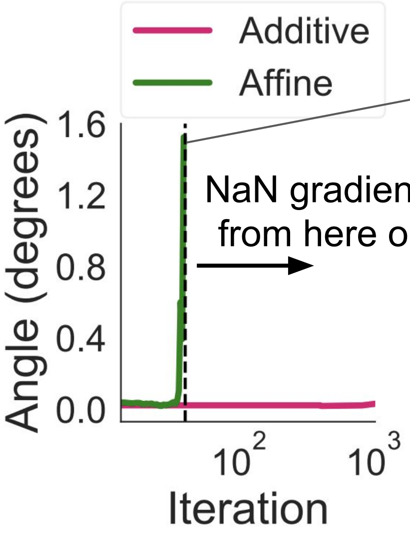
Motivation: Issues with Memory-Saving Gradients

- We can save memory by discarding activations, and *recomputing them in the backward pass, e.g., "memory-saving gradients"*
- Measure the quality of the memory-saving gradient by computing the *angle to the true gradient* (that is computed using stored activations)



Motivation: Issues with Memory-Saving Gradients

- We can save memory by discarding activations, and *recomputing them in the backward pass, e.g., "memory-saving gradients"*
- Measure the quality of the memory-saving gradient by computing the *angle to the true gradient* (that is computed using stored activations)

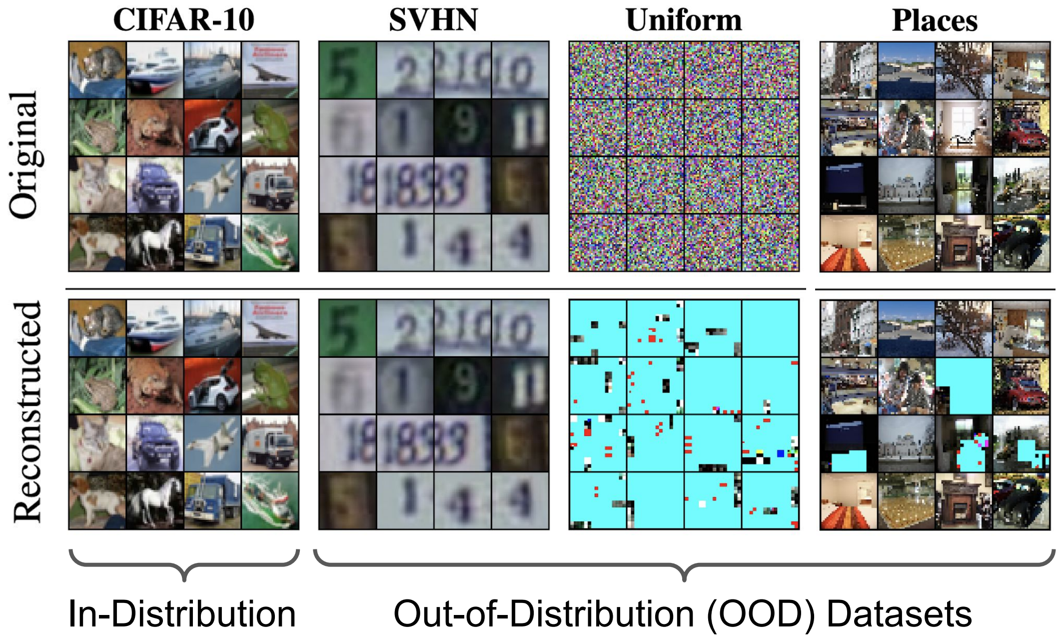


Exploding inverses in affine models lead to highly inaccurate gradients

Foreshadowing: We provide a regularizer that stabilizes affine models and allows for training with memory-saving gradients

Motivation: Instability on OOD Data

- Pre-trained affine Glow models are *not numerically invertible on OOD data!*
  - The exploding inverse will also impact likelihoods on OOD samples, making these models *ill-suited for likelihood-based OOD detection*
- Pre-trained Residual Flows do not suffer from this issue



	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

Tasks Have Different Stability Requirements

- Different tasks have *different stability requirements:*

Memory-Saving Gradients

- Only require the model to be invertible on the training data, to reliably compute gradients

Local stability

Normalizing Flows

- Require the model to be invertible on training and test data, and for many applications on out-of-distribution data

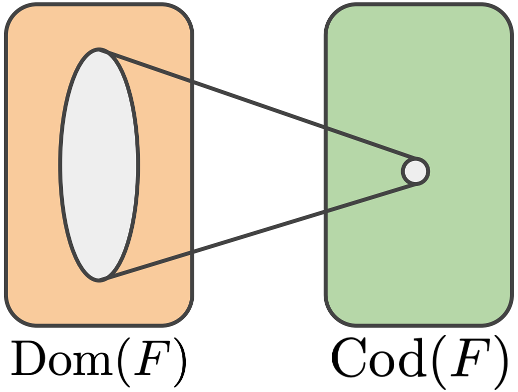
Global stability

Outline

- Motivation
- Lipschitz Properties of INN Building Blocks
- Controlling Global and Local Stability
  - Bi-directional finite differences regularization
  - Normalizing Flow Regularization
- Experiments
  - Instability on OOD Data
  - Non-invertibility within the dequantization region
  - Memory-efficient gradient computation
- Summary and Practical Takeaways

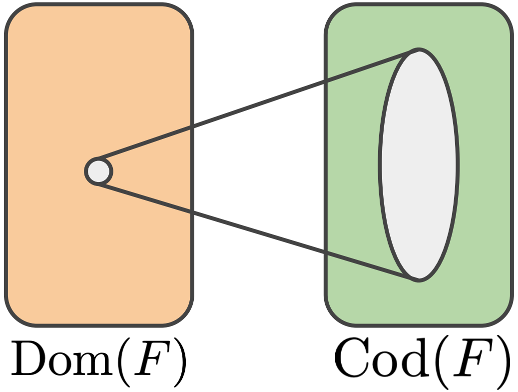
Bi-Lipschitz Continuity

Lipschitz forward ✓
Lipschitz inverse ✗



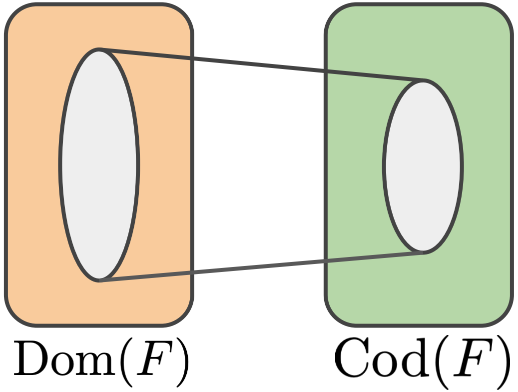
- Small change in input → small change in output

Lipschitz forward ✗
Lipschitz inverse ✓



- Small change in output → small change in input

Lipschitz forward ✓
Lipschitz inverse ✓



- *Bi-Lipschitz continuous functions:* changes bounded in both directions

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:*

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

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

$$\|F^{-1}(y_1) - F^{-1}(y_2)\| \leq L^*\|y_1 - y_2\|, \quad \forall\, y_1, y_2 \in \mathbb{R}^d, \tag{2}$$

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 $[a, b]^d$.*

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 $L =: \operatorname{Lip}(F)$ such that:*

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

If an inverse $F^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ and a constant $L^* =: \operatorname{Lip}(F^{-1})$ exists such that:

$$\|F^{-1}(y_1) - F^{-1}(y_2)\| \leq L^*\|y_1 - y_2\|, \quad \forall\, y_1, y_2 \in \mathbb{R}^d, \tag{2}$$

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 $[a, b]^d$.*

- Computations in deep learning are *carried out in limited precision* *→numerical error is always introduced* in both the forward and inverse passes
- Instability in either pass can amplify the imprecision, *making an analytically-invertible network numerically non-invertible!*

Coupling-Based INNs

Additive Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} + t(x_{I_1})$$

Affine Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} \odot \boxed{g(s(x_{I_1}))} + t(x_{I_1})$$

The difference between these coupling blocks is this scaling

Theorem 1

1. *Affine blocks have strictly larger bi-Lipschitz bounds* than additive blocks
2. There is a *global bi-Lipschitz bound for additive blocks*, but *only local bounds for affine blocks*.

Coupling-Based INNs

Additive Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} + t(x_{I_1})$$

Affine Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} \odot g(s(x_{I_1})) + t(x_{I_1})$$

- Affine blocks can have arbitrarily large singular values in the Jacobian of the inverse mapping
  - We call this *exploding inverses*
  - Thus, they are more likely to be numerically non-invertible than additive blocks
- *Controlling stability requires different approaches for additive vs affine blocks*
  - Additive blocks have global bounds
  - Affine blocks are not globally bi-Lipschitz

Outline

- Motivation
- Lipschitz Properties of INN Building Blocks
- Controlling Global and Local Stability
  - Bi-directional finite differences regularization
  - Normalizing Flow Regularization
- Experiments
  - Instability on OOD Data
  - Non-invertibility within the dequantization region
  - Memory-efficient gradient computation
- Summary and Practical Takeaways

Controlling Global Stability

Additive Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} + \boxed{t(x_{I_1})}$$

Affine Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} \odot g(s(x_{I_1})) + t(x_{I_1})$$

Spectral Normalization

- Can control the Lipschitz constant of t, which guarantees stability
- On the other hand, *spectral normalization does not provide guarantees for affine blocks*, as they are not globally bi-Lipschitz due to the dependence on the range of the inputs x
  - Inputs to the first layer are usually bounded by the nature of the data
  - But obtaining bounds for the intermediate activations is less straightforward

Controlling Global Stability

Additive Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} + t(x_{I_1})$$

Affine Coupling

$$F(x)_{I_1} = x_{I_1}$$
$$F(x)_{I_2} = x_{I_2} \odot \boxed{g(s(x_{I_1}))} + t(x_{I_1})$$

Modified Affine Scaling

- A natural way to increase stability of affine blocks is to consider *different elementwise scaling g*
- Avoiding scaling by small values strongly influences the inverse Lipschitz bound
- **One option:** adapt the sigmoid scaling to output values in a restricted range such as (0.5, 1) rather than (0, 1).
  - This improves stability, but does not completely erase qualitative stability issues

Bi-Directional Finite Differences Regularizer

- Penalty terms on the Jacobian can be used to enforce local stability
- If F is Lipschitz continuous and differentiable, then we have:

$$\mathrm{Lip}(F) = \sup_{x \in \mathbb{R}^d} \underbrace{\|J_F(x)\|_2}_{\substack{\text{Spectral norm of the Jacobian} \\ = \\ \text{The largest singular value}}} = \sup_{x \in \mathbb{R}^d} \sup_{\|v\|_2 = 1} \|J_F(x)v\|_2$$

- We introduce a second approximation using finite differences:

$$\boxed{\sup_{x \in \mathbb{R}^d} \sup_{\|v\|_2 = 1} \|J_F(x)v\|_2 \approx \sup_{x \in \mathbb{R}^d} \sup_{\|v\|_2 = 1} \frac{1}{\varepsilon} \|F(x) - F(x + \varepsilon v)\|_2}$$

Finite Differences Regularization

Influence of Normalizing Flow Loss on Stability

- The training objective itself can impact local stability
- Consider the commonly-used *normalizing flow objective*:

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

- The log-determinant can be expressed as:

$$\log |\det J_{F_\theta}| = \underbrace{\sum_{i=1}^{d} \log \sigma_i(x)}_{\text{Minimizing the NLL invoves minimizing the } \textit{sum of the log singular values}}$$

- Due to the slope of the log function, small singular values are avoided

Influence of Normalizing Flow Loss on Stability

- The training objective itself can impact local stability
- Consider the commonly-used *normalizing flow objective*:

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

- When using $Z \sim \mathcal{N}(0, I)$ as the base distribution, we minimize:

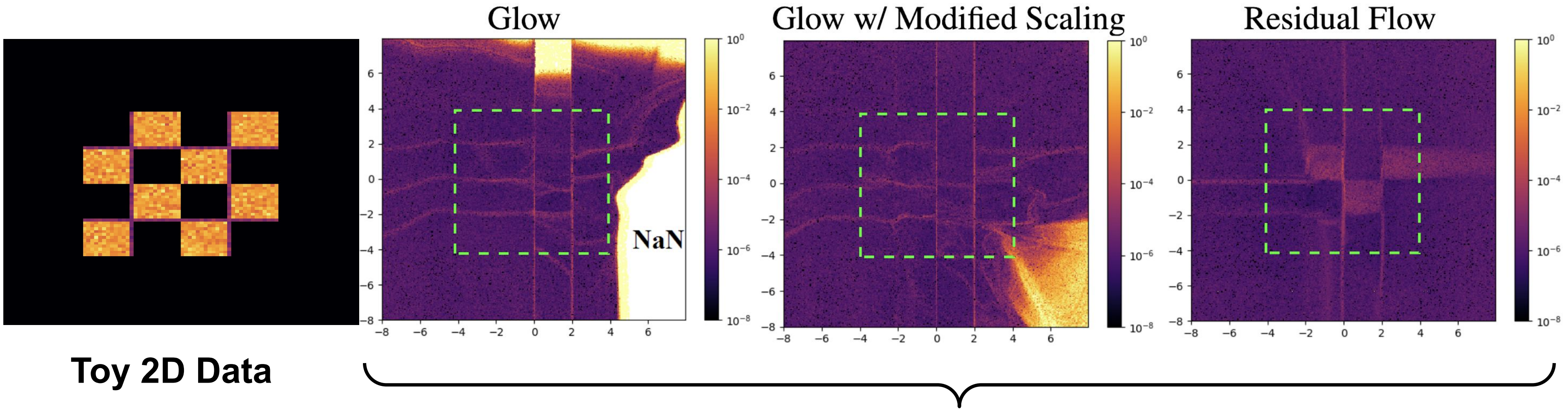
$$-\log p_Z(F_\theta(x)) \propto \|F_\theta(x)\|_2^2$$

- This bounds the L2 norm of the outputs of F
- Avoids large singular values

Outline

- Motivation
- Lipschitz Properties of INN Building Blocks
- Controlling Global and Local Stability
  - Bi-directional finite differences regularization
  - Normalizing Flow Regularization
- Experiments
  - Instability on OOD Data
  - Non-invertibility within the dequantization region
  - Memory-efficient gradient computation
- Summary and Practical Takeaways

Instability on OOD Data



Toy 2D Data

Reconstruction errors for different architectures

- Affine models can become *non-invertible outside the data domain*
- Using modified sigmoid scaling in (0.5, 1) helps stabilize the model, but it still stuffers from exploding inverses in OOD regions
- Residual Flows have *low reconstruction error globally*

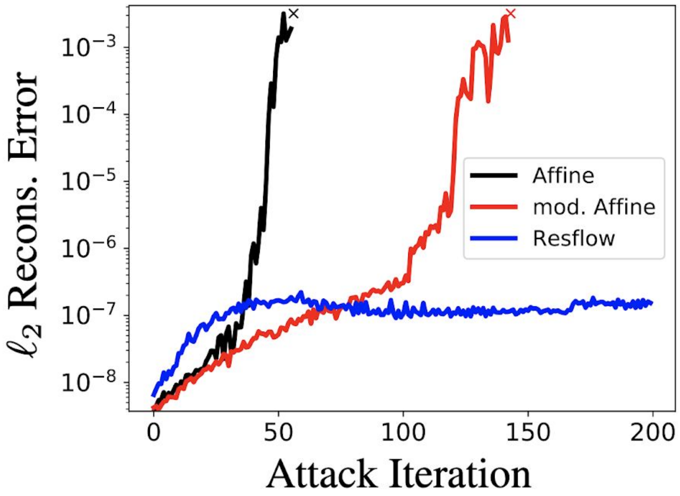
Non-Invertible Inputs within the Dequantization Distribution

- When we train NFs, we *dequantize* the input data x by adding uniform noise $x + \epsilon$

Q: Is it possible to get unlucky when sampling the noise, obtaining a non-invertible $x + \epsilon$?

A: *Yes*, using the invertibility attack we found that there are non-invertible inputs in the dequantization distribution.

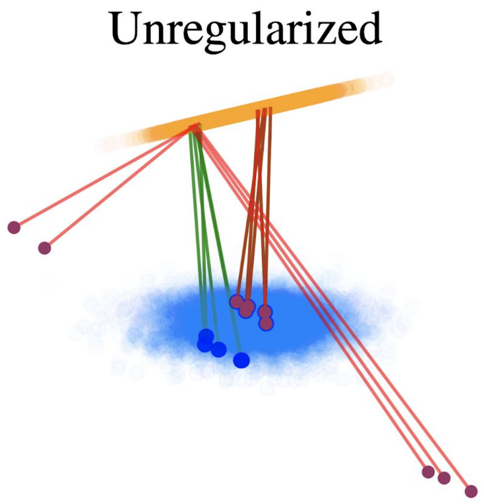
- Sampling such a dequantization may cause training to break



I Find Your Lack of Stability Disturbing

- *No built-in mechanism* to avoid unstable inverses in standard classification/regression

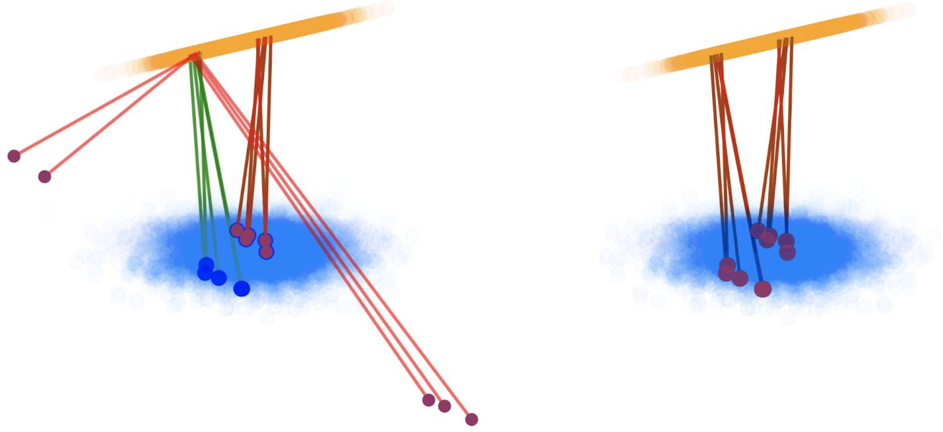
$$\begin{tikzcd} y \arrow[d, bend left, "F^{-1}(y)"] \\ x \arrow[u, bend left, "F(x)"] \end{tikzcd}$$



I Find Your Lack of Stability Disturbing

- *No built-in mechanism* to avoid unstable inverses in standard classification/regression

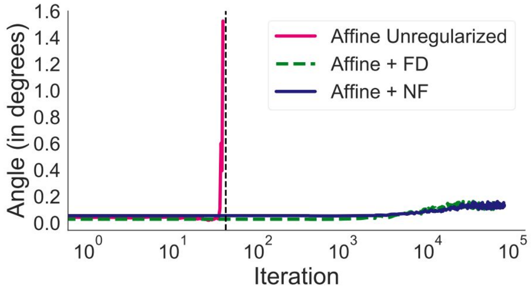
$$
\begin{tikzcd}
y \arrow[d, bend left, "F^{-1}(y)"] \\
x \arrow[u, bend left, "F(x)"]
\end{tikzcd}
$$



- Adding regularization via the *normalizing flow loss with a small coefficient stabilizes the inverse mapping*

Memory-Saving Gradients on CIFAR-10

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



- Additive-coupling models are *numerically stable even without regularization*
- *Unregularized affine models are unstable* due to exploding inverses
    - The singular value of the Jacobian of the inverse mapping is large
- Both *finite-differences and normalizing flow regularizers stabilize the affine model*
    - Reducing the condition number of the mapping

Summary & Practical Takeaways

- INNs enable generative modeling with exact likelihoods and computing memory-saving gradients
  - But the advantages of INNs *rely on the assumption that the models are numerically invertible*
- Tasks have different stability requirements
  - Memory-saving gradients only require local stability on the training data
  - NFs applied to test data & OOD data should ideally be stable globally
- INN architectures have different stability properties
  - Residual Flows are based on stability as a fundamental design principle
  - Additive and affine coupling models have different theoretical properties --- affine models have no global Lipschitz bounds
- Exploding inverses occur when the singular values of the Jacobian of the inverse mapping can become arbitrarily large
- Regularization can be used to stabilize INNs and avoid exploding inverses

Thank you!