Dig deep into INN

Zhiyuan Tang

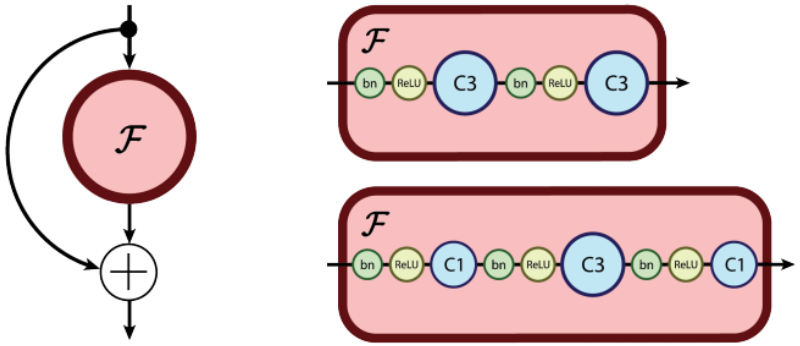
2020.7.13

Question 1

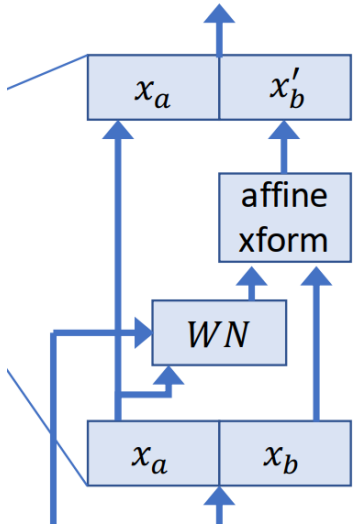
Is invertible neural network (INN) weak?

Residual vs. Coupling layer

$$y = x + \mathcal{F}(x),$$



$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} + m(\mathbf{x}_{1:d}) \end{cases}$$



ResNet vs. INN

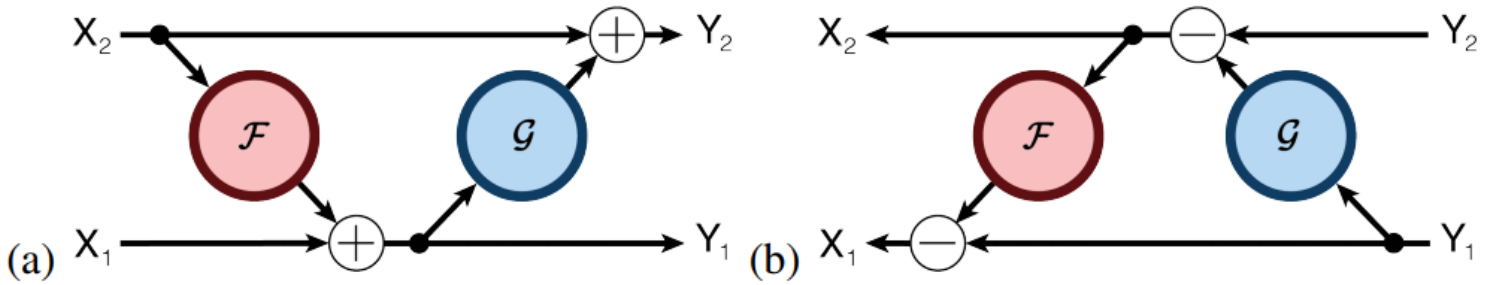


Figure 2: **(a)** the forward, and **(b)** the reverse computations of a residual block, as in Equation 8.

$$y_1 = x_1 + \mathcal{F}(x_2)$$
$$y_2 = x_2 + \mathcal{G}(y_1)$$

The reversible residual network: Backpropagation without storing activations

ResNet vs. INN

Table 1: Computational and spatial complexity comparisons. L denotes the number of layers.

Technique	Spatial Complexity (Activations)	Computational Complexity
Naive	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Checkpointing [22]	$\mathcal{O}(\sqrt{L})$	$\mathcal{O}(L)$
Recursive Checkpointing [5]	$\mathcal{O}(\log L)$	$\mathcal{O}(L \log L)$
Reversible Networks (Ours)	$\mathcal{O}(1)$	$\mathcal{O}(L)$

The reversible residual network: Backpropagation without storing activations

INN

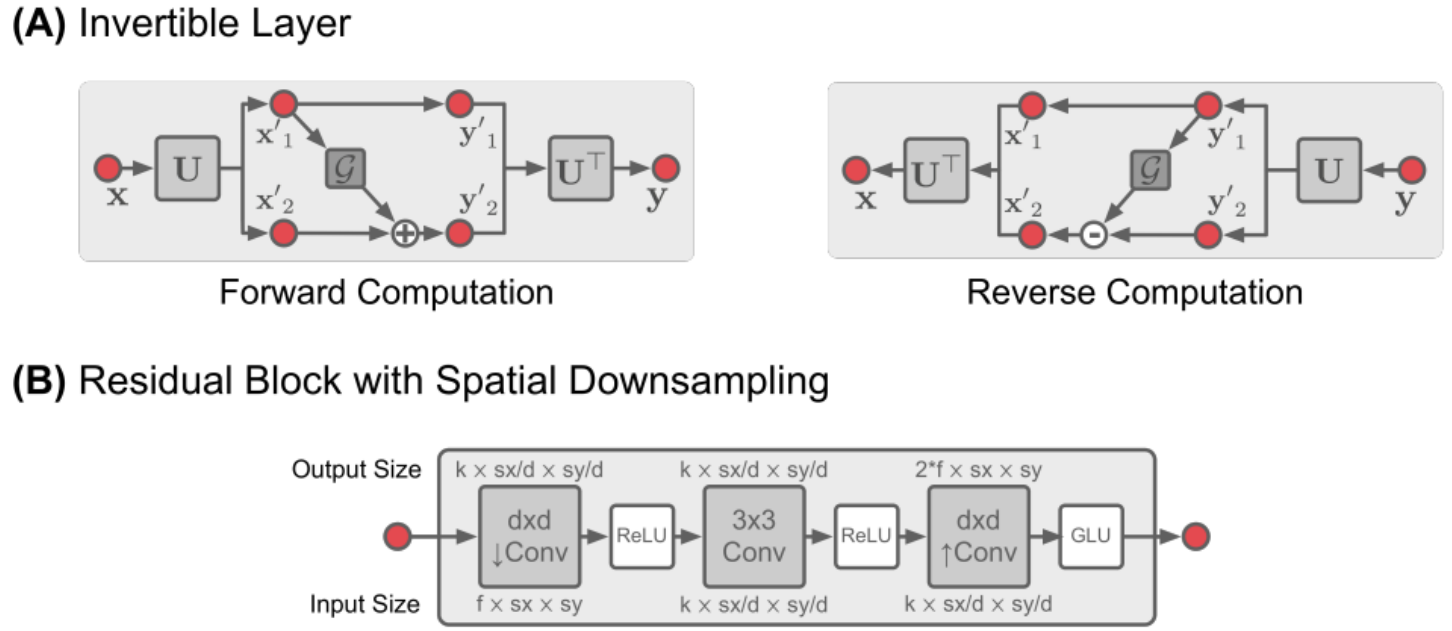
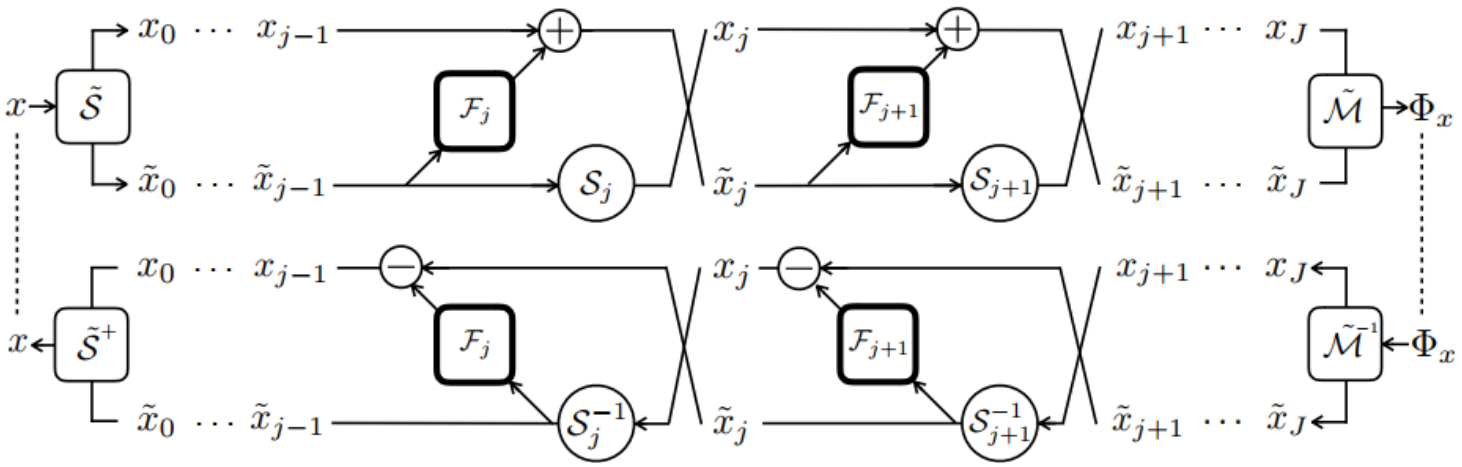


Figure 2: Illustration of the layer used in this work. (A): Invertible layer with orthogonal 1×1 convolutional embedding. (B): Function $\mathcal{G}$ used in each invertible layer.

[Invert to learn to invert]

INN



It is widely believed that the success of deep convolutional networks is based on progressively discarding uninformative variability about the input with respect to the problem at hand. This is supported empirically by the difficulty of recovering images from their hidden representations, in most commonly used network architectures. In this paper we show via a one-to-one mapping that this loss of information is *not* a necessary condition to learn representations that generalize well on complicated problems, such as ImageNet. Via a cascade of homeomorphic layers, we build the i-RevNet, a network that can be fully inverted up to the final projection onto the classes, i.e. no information is discarded. Building an invertible architecture is difficult, for one, because the local inversion is ill-conditioned, we overcome this by providing an explicit inverse. An analysis of i-RevNets learned representations suggests an alternative explanation for the success of deep networks by a progressive contraction and linear separation with depth. To shed light on the nature of the model learned by the i-RevNet we reconstruct linear interpolations between natural image representations.

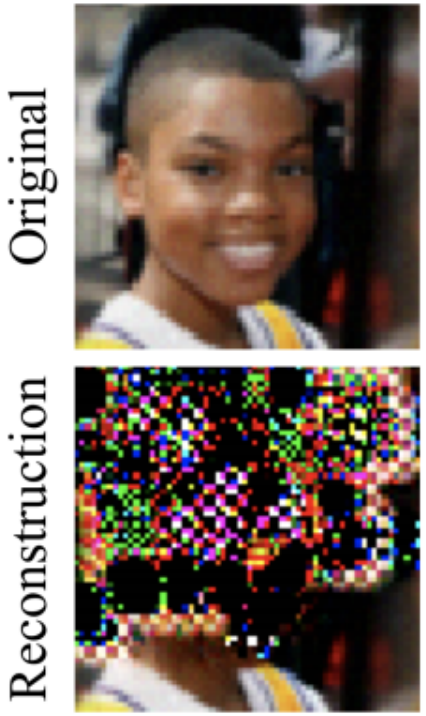
i-revnet: Deep invertible networks

Question 2

- Why is flow's performance weak?
  - Some reason from

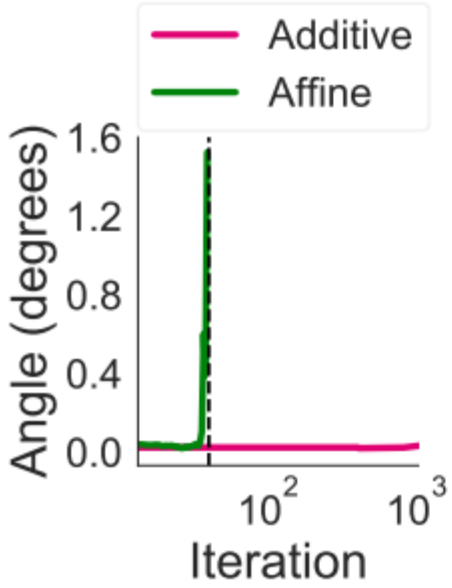
Understanding and mitigating exploding inverses in invertible neural networks

Exploding inverses



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 [33]. 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 [48].

Exploding inverses



Memory-saving gradient computation [22] 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 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. In contrast, memory-saving gradients only require the model to be invertible on the training data, to reliably compute gradients.

Exploding inverses

The proof is given in Appendix B.1, together with upper bounds in Lemmas 5 and 6. Note that an upper bound on $\mathrm{Lip}(F)$ provides a lower bound on $\mathrm{Lip}(F^{-1})$ 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 Lipschitz bounds of other common INN building blocks in Table 2 (Appendix A). We cover coupling-based approaches, free-form approaches like Neural ODEs [10] and i-ResNets [7]. Note that the bounds provide the worst-case stability and are primarily meant to serve as a guideline for the design of invertible blocks.

Exploding inverses

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 [48]. 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:

$$\log p_\theta(x) = \log p_Z(F_\theta(x)) + \log \left|\det J_{F_\theta}(x)\right|, \quad \forall\, x \in \mathbb{R}^d, \tag{6}$$

where $J_{F_\theta}(x)$ denotes the Jacobian of F_θ at x. The log-determinant in Eq. 6 can be expressed as:

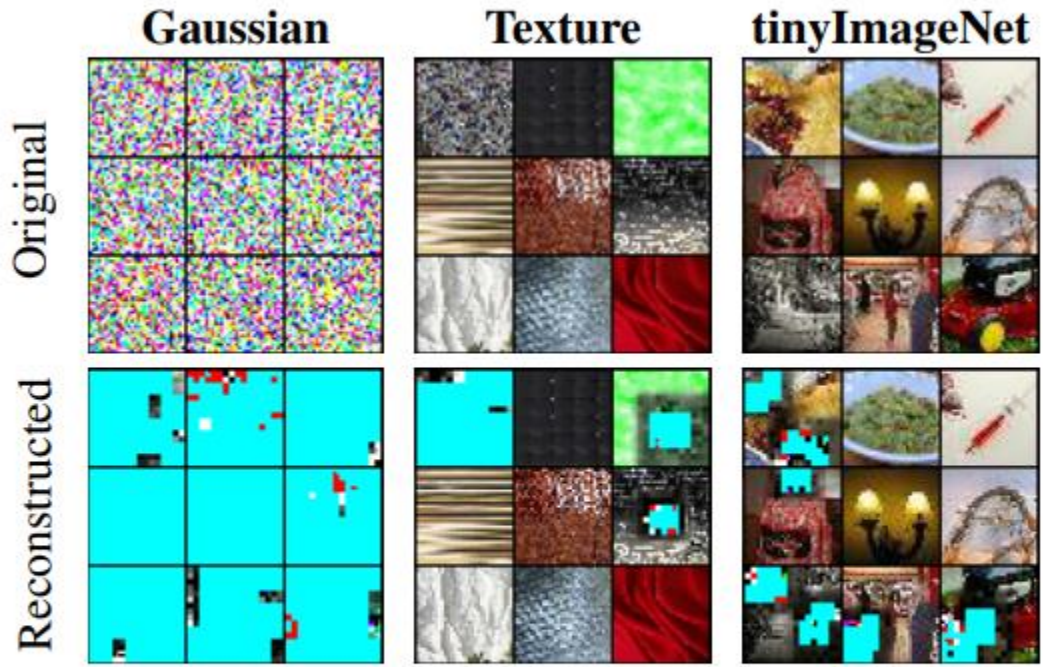
$$\log \left|\det J_{F_\theta}(x)\right| = \sum_{i=1}^{d} \log \sigma_i(x), \tag{7}$$

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 minimizing the sum of the log singular values (Eq. 7). 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_θ is encouraged to be more stable than the forward mapping. Furthermore when using $Z \sim \mathcal{N}(0, I)$ as the base distribution, we minimize:

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

which bounds the ℓ_2-norm of the outputs of F_θ. Due to this effect, large singular values are avoided and the mapping F_θ 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. 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).

Exploding inverses



Dataset	Glow	ResFlow
CIFAR-10 (in-dist)	6.26e-5	2.86e-2
Uniform	Inf	1.69e-2
Gaussian	Inf	7.24e-3
Rademacher	Inf	1.90e-3
SVHN [47]	5.51e-5	7.25e-2
Texture [11]	Inf	2.03e-2
Places [60]	Inf	2.94e-2
tinyImageNet	Inf	3.47e-2

Figure 2: **Left:** Reconstructions of OOD data, using a CIFAR-10 pre-trained Glow model. Broken regions in the reconstructions are plotted in cyan. **Right:** Mean ℓ_2 reconstruction errors on in-distribution (CIFAR-10) and out-of-distribution data, for a pre-trained Glow and Residual Flow.

Exploding inverses

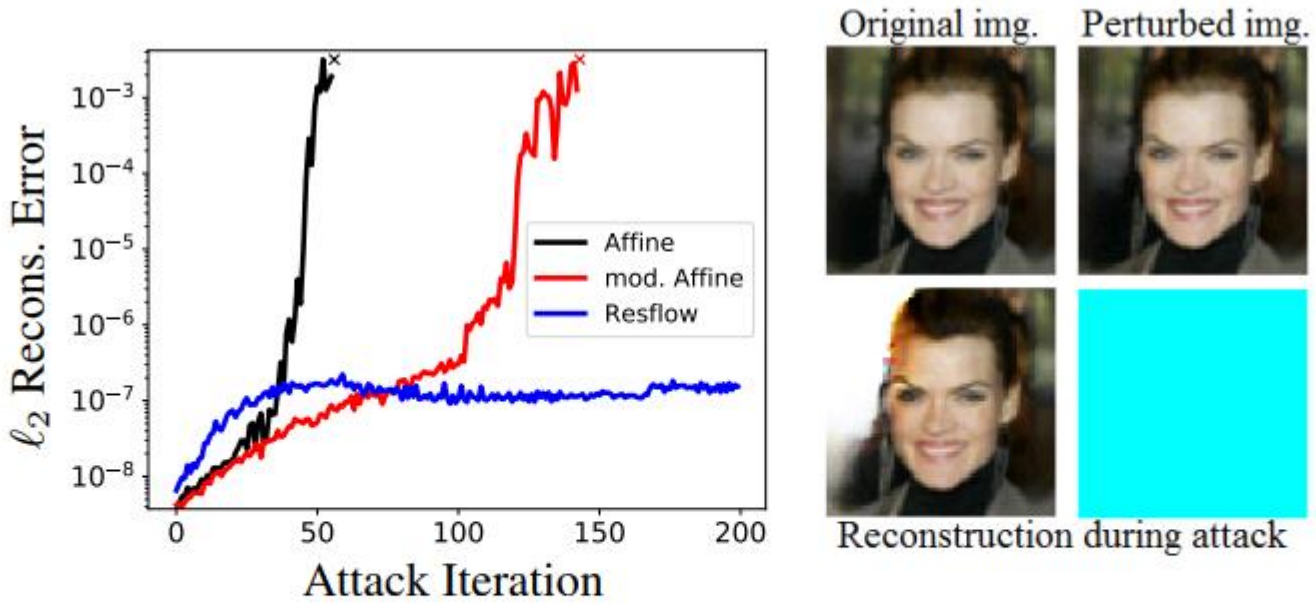


Figure 4: **Non-invertibility within the data-distribution of CelebA64 revealed by an invertibility attack. Left:** ℓ_2-reconstruction error obtained by the attack. The reconstructions of both affine models explode to `NaN` (indicated by $\times$), while the Residual Flow remains stable. **Right:** Despite no visual differences between the original and perturbed image, reconstruction fails for the affine model.

Exploding inverses

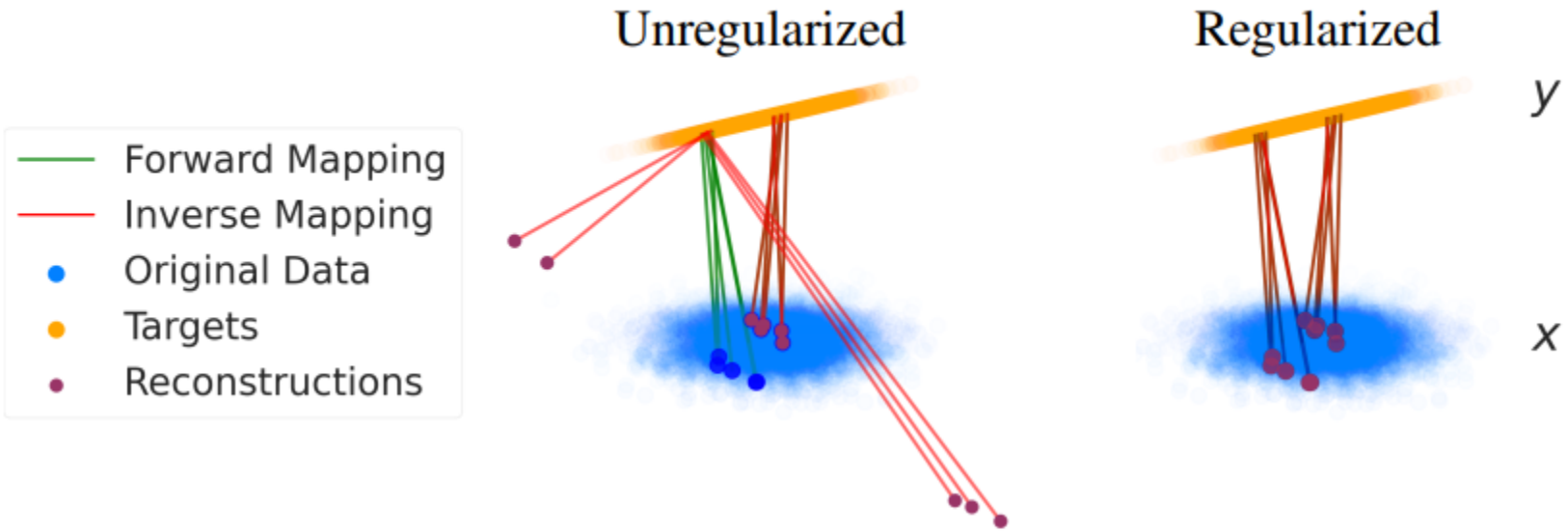


Figure 5: **Visualization of an exploding inverse on a 2D regression task.** A Glow model is trained to map between two 2D Gaussian distributions $(x_1, x_2) \rightarrow (y_1, y_2)$, where y_2 has low variance, so that we are essentially mapping from 2D space onto a 1D subspace. **Left:** An unregularized model suffers from 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.