Inverse Problems with Invertible Neural Networks

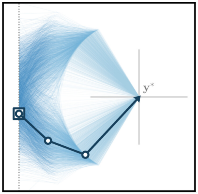
Ardizzone et al., Analyzing Inverse Problems with Invertible Neural Networks, arxiv: 1808.04730

Short Notes

September 2025

Problem statement (1)

- a common scenario: interested in a set of variables $\mathbf{x} \in \mathbb{R}^D$ describing some **phenomenon** of interest ("*primary particle*"), but only variables $\mathbf{y} \in \mathbb{R}^M$ ("*IACT images*") can actually be **observed**
- theory provides a **model** $\mathbf{y} = s(\mathbf{x})$ for the **forward** process
- the task is to determine hidden system parameters $\mathbf{x}$ from a set of measurements $\mathbf{y}$
  - often, the forward process $\mathbf{y} = s(\mathbf{x})$ is well-defined, (*e.g.*, CORSIKA)
  - whereas the inverse problem is ambiguous: multiple parameter sets can result in the same measurement: IACT, HiSCORE(?),

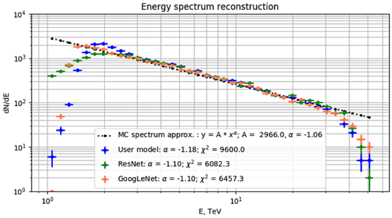
e.g., kinematics 

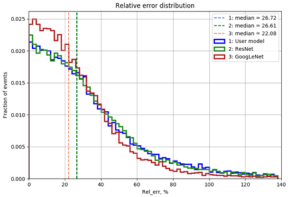
Problem statement (2)

- **classical neural networks** (=regressors/classificators) attempt to solve the ambiguous inverse problem **directly**
- INNs focus on learning the forward process, using additional latent output variables to capture the information otherwise lost
- due to **invertibility**, a model of the corresponding **inverse process** is **learned implicitly**
- given a specific **measurement** and the **distribution of the latent variables**, the inverse pass of the INN provides the **full posterior** over parameter space (e.g., *energy*, etc).
  - INNs are a powerful analysis tool to find **multi-modalities** in parameter space, uncover **parameter correlations**, and identify **unrecoverable** parameters

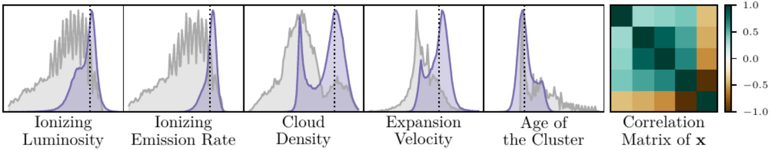
Regressor vs INN: outputs

- Regressor



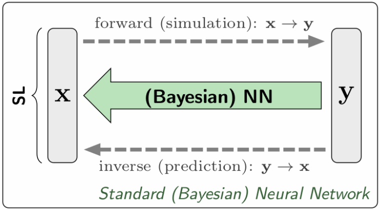


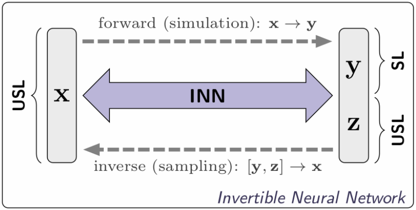
- INN output



Problem statement (3)

- aim at approximating $p(\mathbf{x} \mid \mathbf{y})$ by a tractable model $q(\mathbf{x} \mid \mathbf{y})$, taking advantage of the possibility to create an arbitrary amount of training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ from the known forward model $s(\mathbf{x})$ and a suitable prior $p(\mathbf{x})$.





Problem statement: ingredients

Spaces/Sets

- $\mathbf{x} \in \mathbb{R}^D$ = set of variables describing some phenomenon of interest
- $\mathbf{y} \in \mathbb{R}^M$ variables which can actually be observed
- $\mathbf{z} \in \mathbb{R}^K$ random variable $\mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I_K)$

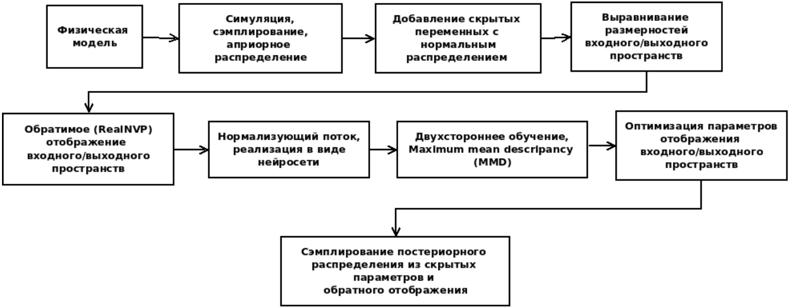
Mappings

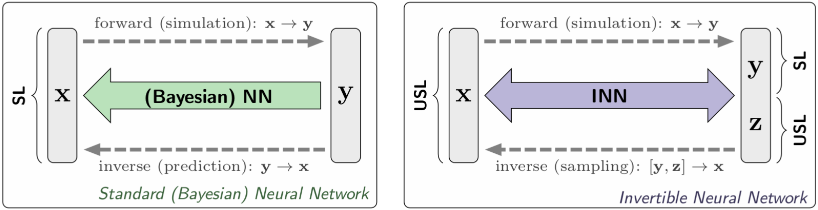
- $\mathbf{y} = s(\mathbf{x})$ **model** provided by the theory = **forward** process
  - $s(\mathbf{x}) \Rightarrow$ an arbitrary amount of training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ (simulations)
- $[\mathbf{y}, \mathbf{z}] = f(\mathbf{x}; \theta) = [f_\mathbf{y}(\mathbf{x}; \theta), f_\mathbf{z}(\mathbf{x}; \theta)] = g^{-1}(\mathbf{x}; \theta)$
- $\mathbf{x} = g(\mathbf{y}, \mathbf{z}; \theta)$

Distros ($q(\cdot)$ = approximations of ...)

- $p(\mathbf{x})$ = prior (simulation inputs)
- $p(\mathbf{y})$ = simulation outcomes
- $p(\mathbf{z}) \sim \mathcal{N}$
- $p(\mathbf{x}|\mathbf{y})$ = posterior (conditional)

General workflow





Method (1)

- introduce a latent random variable $\mathbf{z} \in \mathbb{R}^K$ and reparametrize $q(\mathbf{x} \,|\, \mathbf{y})$ in terms of a **deterministic function** g of $\mathbf{y}$ and $\mathbf{z}$, represented by a **neural network** with parameters θ:

$$\mathbf{x} = g(\mathbf{y}, \mathbf{z}; \theta) \quad \text{with} \quad \mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I_K). \tag{1}$$

- learn the model $g(\mathbf{y}, \mathbf{z}; \theta)$ of the inverse process **jointly** with a model $f(\mathbf{x}; \theta)$ approximating the known forward process $s(\mathbf{x})$:

$$[\mathbf{y}, \mathbf{z}] = f(\mathbf{x}; \theta) = [f_{\mathbf{y}}(\mathbf{x}; \theta), f_{\mathbf{z}}(\mathbf{x}; \theta)] = g^{-1}(\mathbf{x}; \theta) \tag{2}$$
$$\text{with} \quad f_{\mathbf{y}}(\mathbf{x}; \theta) \approx s(\mathbf{x}).$$

Functions f and g share the **same parameters** θ and are implemented by a **single invertible neural network**.

Method (2)

- The relation $f = g^{-1}$ is enforced by the invertible network architecture, provided that the nominal and intrinsic dimensions of both sides match.
- When $m \leq M$ denotes the intrinsic dimension of $\mathbf{y}$ ($\sim$ Hillas!), the latent variable $\mathbf{z}$ must have dimension $K = D - m$, assuming that the intrinsic dimension of $\mathbf{x}$ equals its nominal dimension D.
- If the resulting nominal output dimension $M + K$ exceeds D, we augment the input with a vector $\mathbf{x}_0 \in \mathbb{R}^{M+K-D}$ of zeros and replace $\mathbf{x}$ with the concatenation $[\mathbf{x}, \mathbf{x}_0]$ everywhere.

Method (3)

- Thus, our INN learns to associate hidden parameter values $\mathbf{x}$ with unique pairs $[\mathbf{y}, \mathbf{z}]$ of measurements and latent variables.
- Forward training optimizes the mapping $[\mathbf{y}, \mathbf{z}] = f(\mathbf{x})$ and implicitly determines its inverse $\mathbf{x} = f^{-1}(\mathbf{y}, \mathbf{z}) = g(\mathbf{y}, \mathbf{z})$.
- Additionally, we make sure that the density $p(\mathbf{z})$ of the latent variables is shaped as a Gaussian distribution.
- Combining these definitions, our network expresses $q(\mathbf{x} \mid \mathbf{y})$ as

$$q\big(\mathbf{x} = g(\mathbf{y}, \mathbf{z}; \theta) \mid \mathbf{y}\big) = p(\mathbf{z}) \left| J_{\mathbf{x}} \right|^{-1}, \quad J_{\mathbf{x}} = \det\left(\left. \frac{\partial g(\mathbf{y}, \mathbf{z}; \theta)}{\partial [\mathbf{y}, \mathbf{z}]} \right|_{\mathbf{y}, f_{\mathbf{z}}(\mathbf{x})} \right) \tag{3}$$

  with Jacobian determinant $J_{\mathbf{x}}$.
- Thus, the INN represents the desired posterior $p(\mathbf{x} \mid \mathbf{y})$ by a deterministic function $\mathbf{x} = g(\mathbf{y}, \mathbf{z})$ that transforms ("pushes") the known distribution $p(\mathbf{z})$ to $\mathbf{x}$-space, conditional on $\mathbf{y}$.

Invertible architecture

affine coupling layer f: $\mathbf{x} \mapsto \mathbf{y}$:

$$\begin{aligned} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(\sigma(\mathbf{x}_{1:d})) + \mu(\mathbf{x}_{1:d}) \end{aligned} \tag{4}$$

where $\sigma(.)$ and $\mu(.)$ are scale and translation functions and both map $\mathbb{R}^d \mapsto \mathbb{R}^{D-d}$. The operation $\odot$ is the element-wise product.

- easily invertible

$$\begin{cases} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(\sigma(\mathbf{x}_{1:d})) + \mu(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \tag{5}$$

$$\Leftrightarrow \begin{cases} \mathbf{x}_{1:d} &= \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} &= (\mathbf{y}_{d+1:D} - \mu(\mathbf{y}_{1:d})) \odot \exp(-\sigma(\mathbf{y}_{1:d})) \end{cases} \tag{6}$$

Блок-схема нормализующих потоков

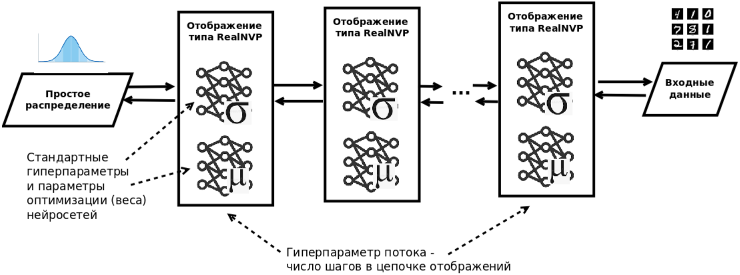
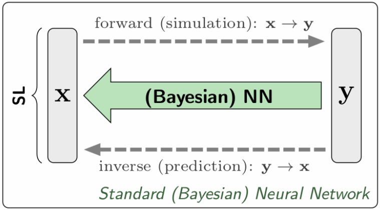
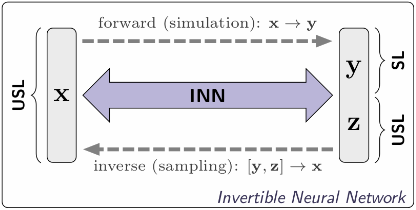


Рис.: Блок-схема нормализующих потоков для моделей типа RealNVP





Bi-directional training (1)

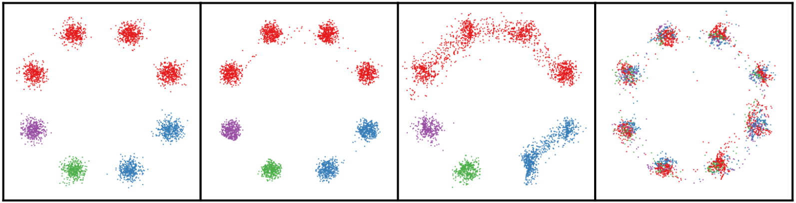
- **forward** and **backward** iterations in an **alternating fashion**, accumulating gradients from both directions before performing a parameter update.
- For the **forward iteration**:
  1. $\mathbf{y} \Rightarrow$ penalize deviations between simulation outcomes $\mathbf{y}_i = s(\mathbf{x}_i)$ and network predictions $f_\mathbf{y}(\mathbf{x}_i)$ with a loss $\mathcal{L}_\mathbf{y}\big(\mathbf{y}_i, f_\mathbf{y}(\mathbf{x}_i)\big) =$ any supervised loss, e.g. squared loss for regression or cross-entropy for classification.
  2. $\mathbf{z} \Rightarrow$ penalize the mismatch between the joint distribution of network outputs $q\big(\mathbf{y} = f_\mathbf{y}(\mathbf{x}), \mathbf{z} = f_\mathbf{z}(\mathbf{x})\big) = p(\mathbf{x})/|J_{\mathbf{yz}}|$ and the product of marginal distributions of simulation outcomes $p\big(\mathbf{y} = s(\mathbf{x})\big) = p(\mathbf{x})/|J_s|$ and latents $p(\mathbf{z})$ as $\mathcal{L}_\mathbf{z}\big(q(\mathbf{y},\mathbf{z}), p(\mathbf{y})\,p(\mathbf{z})\big) =$ **MMD**:
     - $\mathbf{z}$ must follow the desired normal distribution $p(\mathbf{z})$;
     - $\mathbf{y}$ and $\mathbf{z}$ must be independent upon convergence (i.e. $p(\mathbf{z} \,|\, \mathbf{y}) = p(\mathbf{z})$)
     - block the gradients of $\mathcal{L}_\mathbf{z}$ with respect to $\mathbf{y}$ to ensure the resulting updates only affect the predictions of $\mathbf{z}$ and do not worsen the predictions of $\mathbf{y}$.

Bi-directional training (2)

Theorem: *If an INN* $f(\mathbf{x}) = [\mathbf{y}, \mathbf{z}]$ *is trained as proposed, and both the supervised loss* $\mathcal{L}_{\mathbf{y}} = \mathbb{E}[(\mathbf{y} - f_{\mathbf{y}}(\mathbf{x}))^2]$ *and the unsupervised loss* $\mathcal{L}_{\mathbf{z}} = D\big(q(\mathbf{y}, \mathbf{z}), p(\mathbf{y})\, p(\mathbf{z})\big)$ *reach zero, sampling according to Eq. 1 with* $g = f^{-1}$ *returns the true posterior* $p(\mathbf{x} \,|\, \mathbf{y}^*)$ *for any measurement* $\mathbf{y}^*$*.*

- $\mathcal{L}_{\mathbf{y}}$ and $\mathcal{L}_{\mathbf{z}}$ are sufficient asymptotically **but** a small amount of residual dependency between **y** and **z** remains after a finite amount of training $\Rightarrow$
- **backward** iterations = loss $\mathcal{L}_{\mathbf{x}}$ = MMD.
  - matches the distribution of backward predictions $q(\mathbf{x}) = p\big(\mathbf{y} = f_{\mathbf{y}}(\mathbf{x})\big)\, p\big(\mathbf{z} = f_{\mathbf{z}}(\mathbf{x})\big) / |J_{\mathbf{x}}|$ against the prior data distribution $p(\mathbf{x})$ through $\mathcal{L}_{\mathbf{x}}\big(p(\mathbf{x}), q(\mathbf{x})\big)$.
  - proved: $\mathcal{L}_{\mathbf{x}}$ is guaranteed to be zero when the forward losses $\mathcal{L}_{\mathbf{y}}$ and $\mathcal{L}_{\mathbf{z}}$ have converged to zero.
  - $\mathcal{L}_{\mathbf{x}}$ does not alter the optimum, but improves convergence
- control for padded dimensions

Bi-directional training (2)

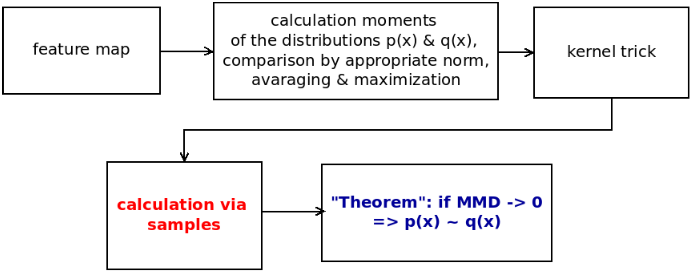


Ground truth INN, all losses INN, only $\mathcal{L}_{\mathbf{y}} + \mathcal{L}_{\mathbf{z}}$ INN, only $\mathcal{L}_{\mathbf{x}}$

- The task is to produce the correct (multi-modal) distribution of 2D points $\mathbf{x}$, given only the color label $\mathbf{y}^*$.
- When trained with all loss terms, the INN output matches ground truth almost exactly *(2nd image)*.
- The ablations *(3rd and 4th image)* show that we need $\mathcal{L}_{\mathbf{y}}$ and $\mathcal{L}_{\mathbf{z}}$ to learn the conditioning correctly
- whereas $\mathcal{L}_{\mathbf{x}}$ helps us remain faithful to the prior.

Maximum mean discrepancy (MMD)

- MMD is a kernel-based method for comparison of two probability distributions that are only accessible **through samples**
- some rather profound math, e.g., feature map, reproducing kernel Hilbert space (RKHS), kernel method + kernel trick,...



- Since the magnitude of the MMD depends on the kernel choice, the relative weights of the losses $\mathcal{L}_{\mathbf{x}}$, $\mathcal{L}_{\mathbf{y}}$, $\mathcal{L}_{\mathbf{z}}$ are adjusted as hyperparameters, such that their effect is about equal.

Вместо заключения (1)

Регрессор	INN
прямое сопоставление (степень похожести на тренировочный экземпляр)	используют все априорное распределение данных для вывода решений
результат: предсказание конкретного значения физ. параметра	**результат**: все апостериорное распределение, в том числе корреляции между физическими параметрами
единственное решение обратной задачи	генерируют несколько разнообразных решений (мультимодальное постериорное распределение), способны справляться с присущей обратным задачам неоднозначностью.

Вместо заключения (2)

- Совсем не рассматривались подходы к решению обратных задач на основе других генеративных моделей
  - в частности, авторы утверждают, что INN позволяет обучаться прямому процессу и получать (**более сложный**) обратный процесс "бесплатно", в отличие, например, от **cGAN**, которые фокусируются на обратном процессе и обучаются прямому процессу лишь неявно.
- Рассмотренная пионерская ($>$ 700 цитирований) работа – довольно старая (2019г.) . С тех пор направление постоянно развивалось, появилось много работ: как теоретических (мат. основы), так и прикладных.