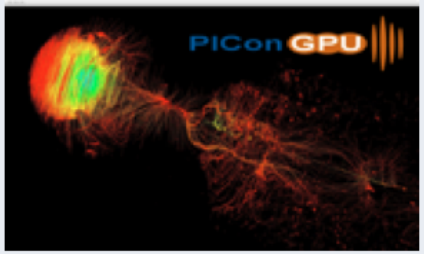
Analyzing Inverse Problems with Invertible Neural Networks

Nico Hoffmann

15th December 2020

YIG: AI for Future Photon Science

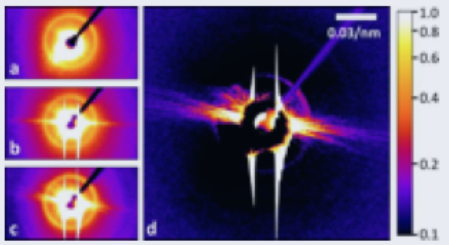
Evolution of that system



Description of a physical system

$$i\frac{\delta\psi}{\delta t} = \frac{1}{2}(-\nabla^2 + \cdots)$$

Experimental exploration



(image: Kluge et al., 2017)

DRESDEN concept

HZDR

That's us!








YOU! We are hiring PhD students + PostDocs!

=> **http://bit.do/joinHZDR**

DRESDEN concept

HZDR

Problem Introduction

For many applications, especially complex systems, the forward process loses some crucial information making the inverse process ill-posed. That means the inverse process is uncertain, i.e. multiple variables x can result in the same measurement y.

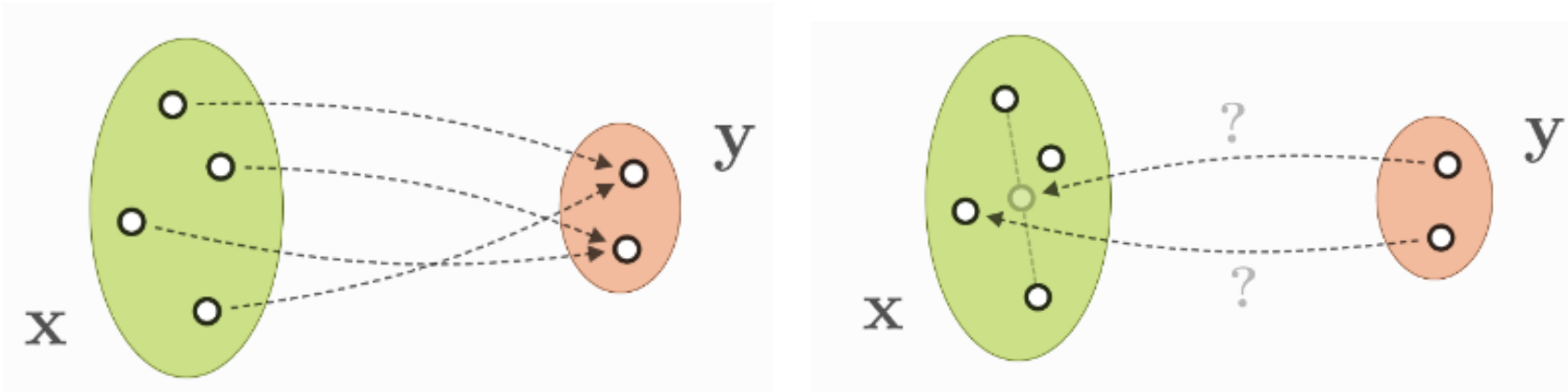
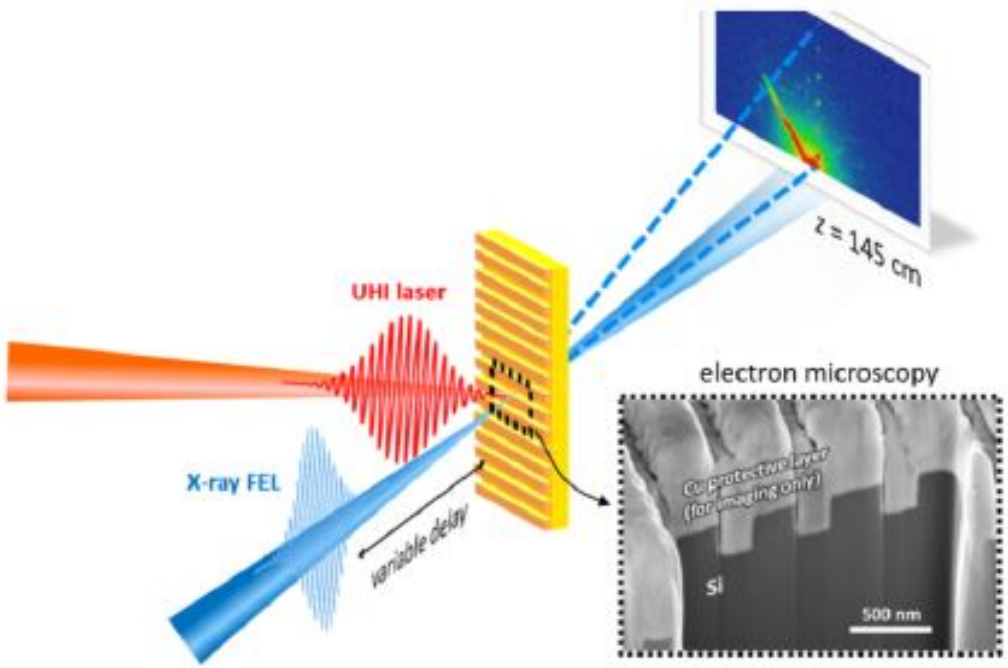
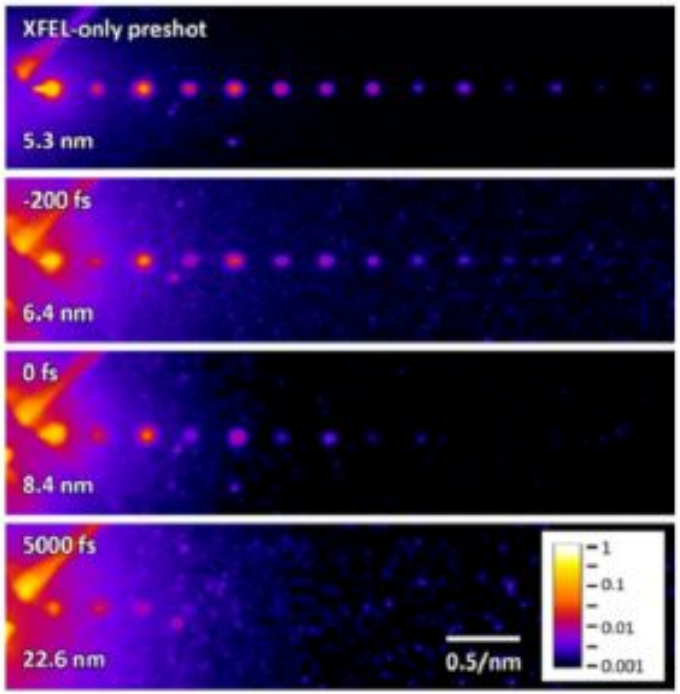


Figure: The intrinsic dimension of **observation** y is typically lower than independent variables x resulting in an ambigous inverse problem.

Example: Small-angle X-ray scattering

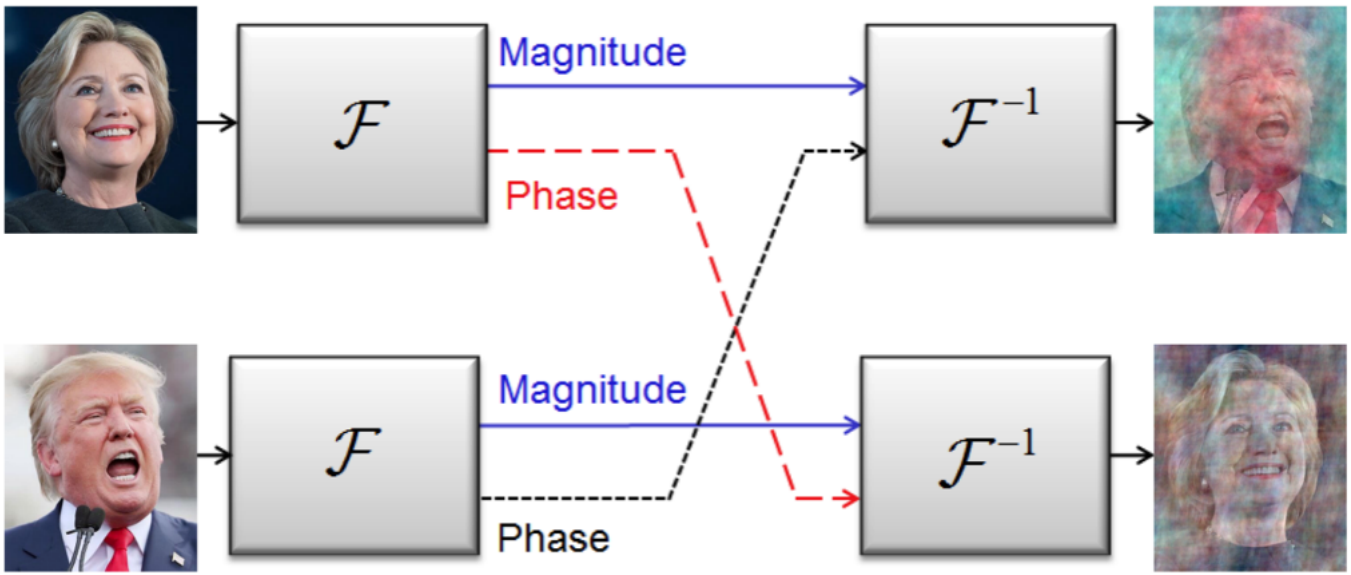




DRESDEN concept

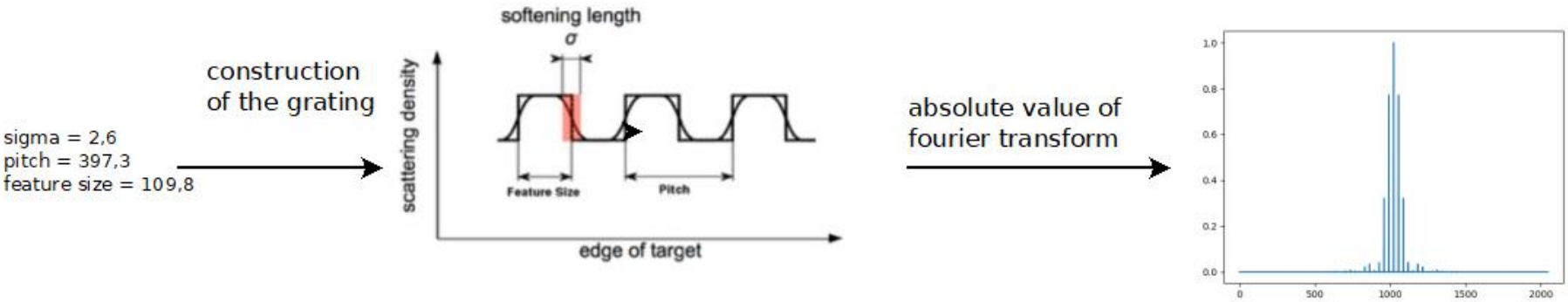
HZDR

Phase Problem

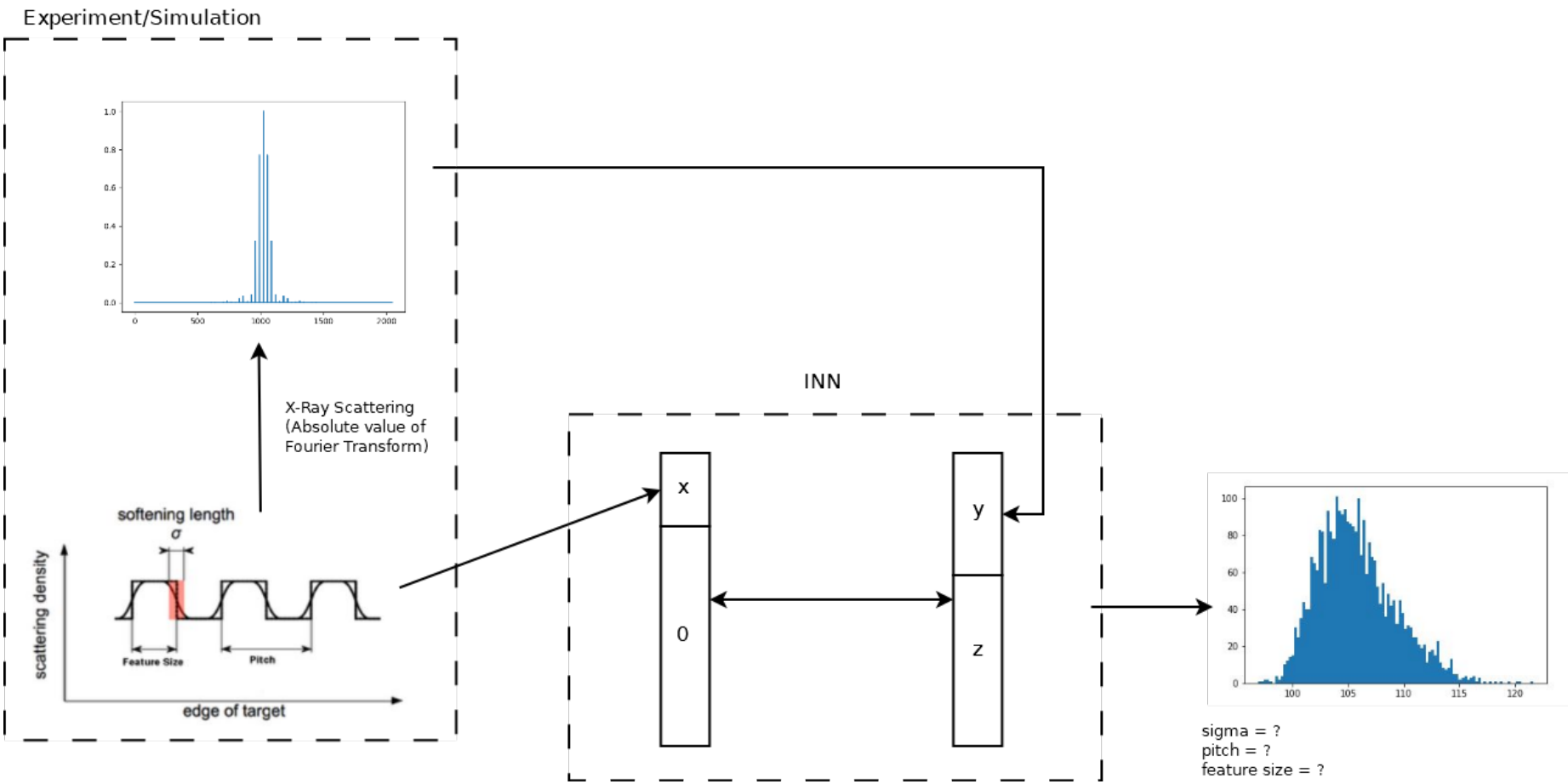


Forward & inverse process

Formation of diffraction data (here: 1D line-out) can be described by analytic model of **forward process** (σ, pitch, feature size).
Inverse process denotes reconstruction of model parameters based on experimentally acquired data.



Reconstruction of SAXS data using invertible neural networks



Normalizing flow 101

A **normalizing flow** renders the data y as output of a *invertible* function $f(z) = y$ of randomly sampled noise $z \sim \pi(z)$.

Change of variables now yields posterior predictictive distribution $p(x|y)$ provided f and $\pi(z)$:

$$p(x|y) = \pi(z) \left| det \left(\frac{\delta f^{-1}(y,z)}{\delta [y,z]} \right) \right|^{-1} \tag{1}$$

General idea: the Jacobian determinent (depending on f) basically compresses and expands some noise distribution $\pi(z)$, e.g. $z \sim N(0,1)$.

Normalizing flow 101

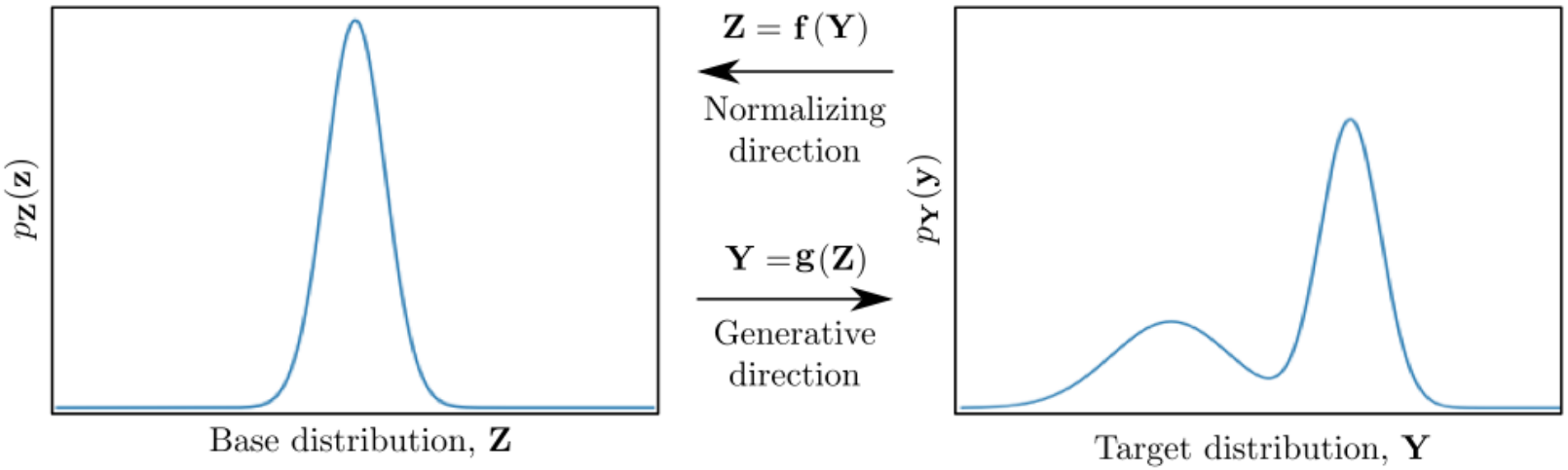
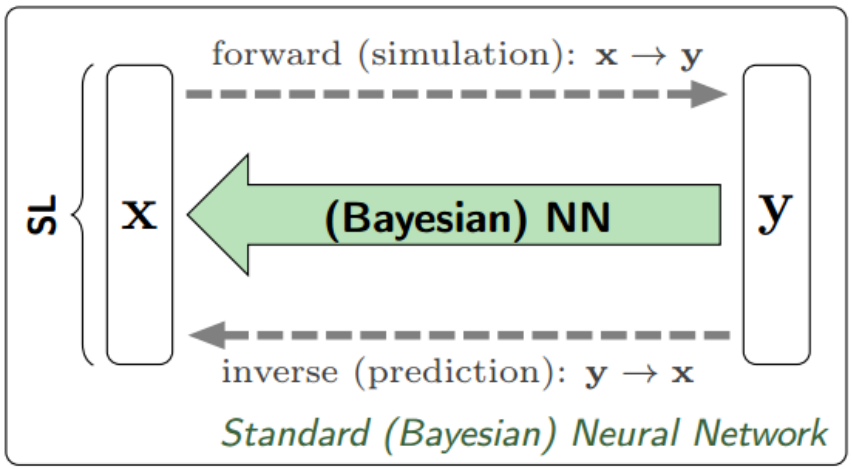
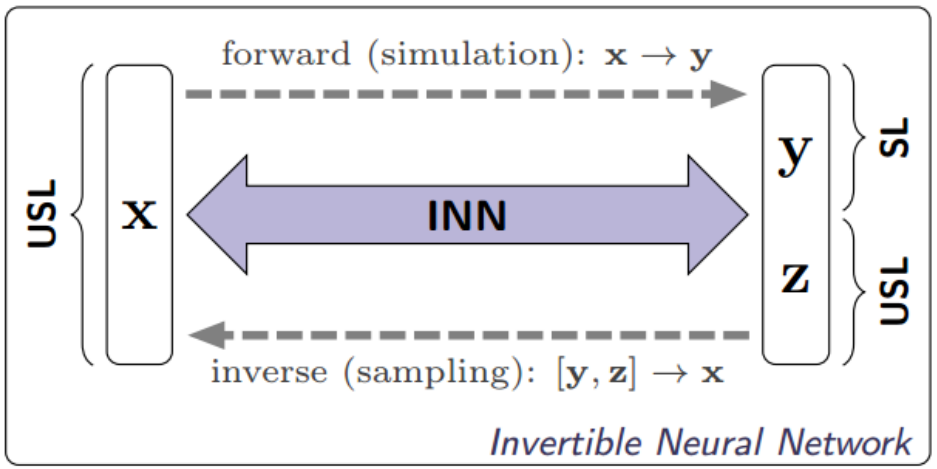


Figure: Mapping from normal distribution $\pi(Z)$ to target distribution $\pi(Y)$ via *unconditioned* normalizing flow. Image source: [Kobyzev et al., 2019]

Solving inverse problems by conditional normalizing flows.

Let's assume there is a latent space $z \in \mathbb{R}^K$ capturing *all* information not contained in $y \in \mathbb{R}^M$. Hereby, the mapping from z, y to $x \in \mathbb{R}^D$ becomes bijective and the inverse function exists.





Problem Specification (compressed)

By definition of the neural network we have that $[\mathbf{y}, \mathbf{z}] = INN(\mathbf{x}; \theta)$ implying: $INN_y(\mathbf{x}; \theta) \approx s(\mathbf{x})$

The INN therefore jointly approximates the

forward pass $s(\mathbf{x}) \approx INN(\mathbf{x}; \theta) = [y, z]$ and

backward pass $INN^{-1}(\mathbf{y}, \mathbf{z}; \theta) = x$

by incorporating simple neural network into coupling transforms.

Bi-directional Training

Loss $\mathcal{L} = \lambda_y \mathcal{L}_\mathbf{y} + \lambda_z \mathcal{L}_\mathbf{z} + \lambda_x \mathcal{L}_\mathbf{x}$.

$\mathcal{L}_\mathbf{y}$: forward pass

$\mathcal{L}_\mathbf{x}$: inverse pass

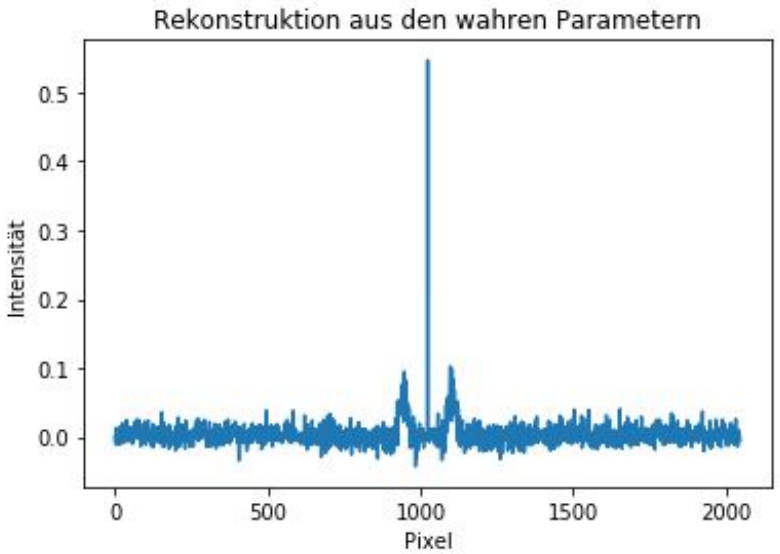
$\mathcal{L}_\mathbf{z}$: independence of z, y

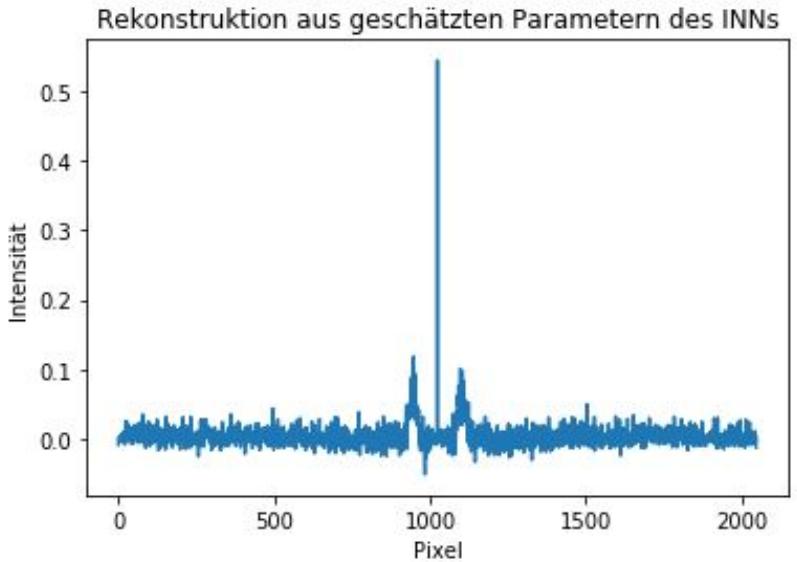
Theorem: *If an INN* $f(\mathbf{x}) = [\mathbf{y}, \mathbf{z}]$ *is trained as proposed, and both the supervised loss* $\mathcal{L}_\mathbf{y} = \mathbb{E}[(\mathbf{y} - f_\mathbf{y}(\mathbf{x}))^2]$ *and the unsupervised loss* $\mathcal{L}_\mathbf{z} = D\big(q(\mathbf{y}, \mathbf{z}), p(\mathbf{y})\,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

Note: this loss can be simplified into single Maximum Likelihood term.

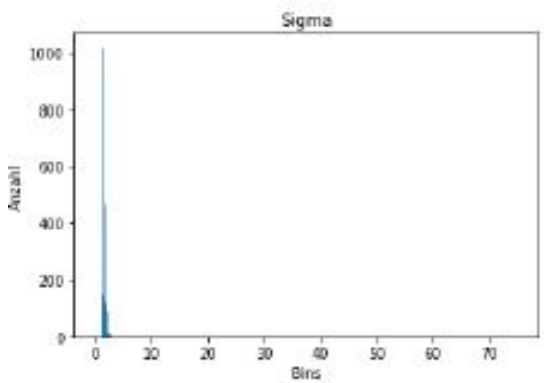
Results

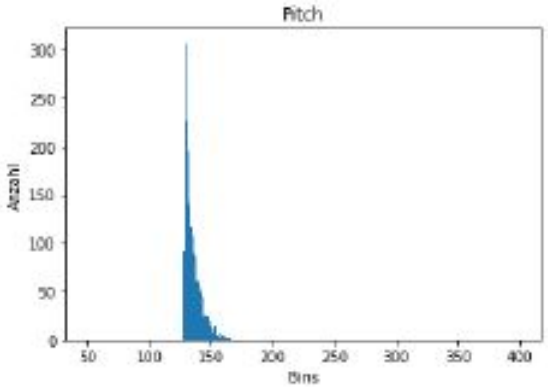
Forward process:

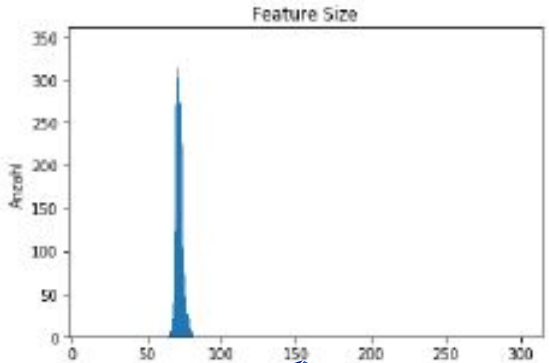




Inverse problem (reconstruction):







DRESDEN concept

HZDR

Summary

- cross-domain surrogate modelling (e.g.Matter, Geophysics, Medical Imaging, Comp. Psychiatry)
- multi-modal data reconstruction
- large-scale ML (Horovod)
- open-source framework ML4IP for solving inverse problems going public, soon. (joint effort with Peter Steinbach's AI consultants)

Thank you!








YOU! We are hiring PhD students + PostDocs!

=> **http://bit.do/joinHZDR**

DRESDEN concept

HZDR

Taxonomy

Residual flows basically improve memory efficiency of *traditional* methods such as UNet, Resnet. (e.g. iRevNet, iUnet). The determinant is difficult to compute :-(

Infinitetesimal flows model residual flows in terms of ordinary/stochastic differential equations (e.g. FFJORD).

Coupling transforms characterize flows that have an analytic inverse but are less flexible (e.g. NICE, RealNVP, GLOW).

Autoregressive flows output y_i depends on x_j (and sometimes y_j) with $j \leq i$. These methods are D times slower to invert than to evaluate $x \in \mathbb{R}^D$. The inverse is computed by numerical methods.

Coupling blocks

Let $\phi : x \to y$ be a coupling transform mapping x onto y. A coupling layer follows four steps:

1. $x \in \mathbb{R}^D$ is splitted into two parts: $[x_1, ..., x_d - 1]$ and $[x_d, ..., x_D]$
2. $\theta = NN([x_1, ..., x_d - 1])$
3. $y_j = x_j$ for $j = 1, \cdots, d-1$
4. $y_i = g_\Theta(x_i)$ for $i = d, \cdots, D$ (g is **elementwise transform** with parameters θ))

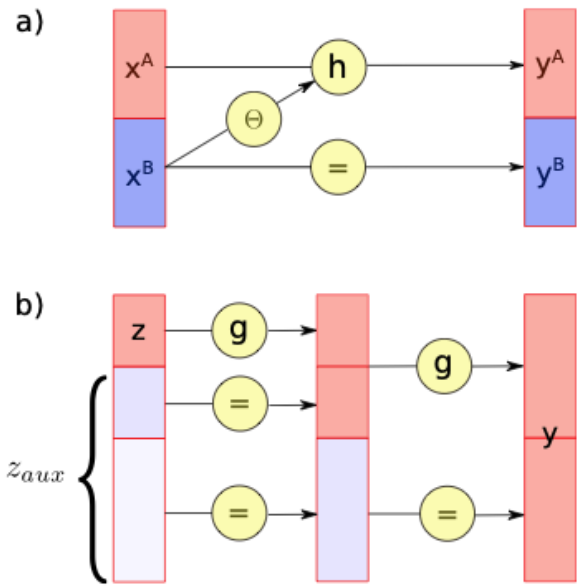


Figure: Sketch of certain coupling architectures (image source: Kobyzev et al.

Elementwise transforms

Affine functions:

$$g_{\theta i}(x_i) = \alpha_i x_i + \beta_i \;\; \texttt{with} \;\; \theta_i = \alpha_i, \beta_i \tag{2}$$

Piecewise-linear/quadratic polynomials with ξ being the relative position of x within bin b from V_b to V_{b+1}.

$$g_\theta(x) = \frac{1}{2}\xi^2(V_b - V_{b+1}) + \xi V_b + \frac{1}{2}\sum_{i=0}^{b-1}(V_k - V_{k+1}) \tag{3}$$

Cubic splines are evaluated at k knots V_k by computing $\xi = x - V_k$. $k = \text{argmin}_k |x - V_k|$ denotes the *closest* knot.

$$g_\theta(\xi)^k = \alpha_{k_0} + \alpha_{k_1}\xi + \alpha_{k_2}\xi^2 + \alpha_{k_3}\xi^3 \tag{4}$$

Problem Specification (1)

Definition: parameters $\mathbf{x} \in \mathbb{R}^D$, observations $\mathbf{y} \in \mathbb{R}^M$, iid. K-dimensional random variable $\mathbf{z} \sim N(0, I_K)$.

Forward process $s(x)$ of parameters x yields measurement y, i.e. $\mathbf{y} = s(\mathbf{x})$.

The **likelihood** function reads: $p(\mathbf{y}|\mathbf{x})$ while the **prior** distribution is $p(x)$. We are now looking for the posterior predictive distribution:

$$p(\mathbf{x}|\mathbf{y}) \approx p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \tag{5}$$

Problem Specification (2)

Inverse problem: $\mathbf{x} = INN^{-1}(\mathbf{y}, \mathbf{z}; \theta)$ with $\mathbf{z} \sim \pi(\mathbf{z}) = \mathcal{N}(0, I_K)$
(θ are parameters of our neural network)

The INN now becomes a tractable model $q(\mathbf{x}|\mathbf{y}) = INN^{-1}(\mathbf{y}, \mathbf{z})$ which approximates the posterior: $p(\mathbf{x}|\mathbf{y}) \approx q(\mathbf{x}|\mathbf{y})$.

By definition of the neural network we also have that $[\mathbf{y}, \mathbf{z}] = INN(\mathbf{x}; \theta)$ implying: $INN_y(\mathbf{x}; \theta) \approx s(\mathbf{x})$

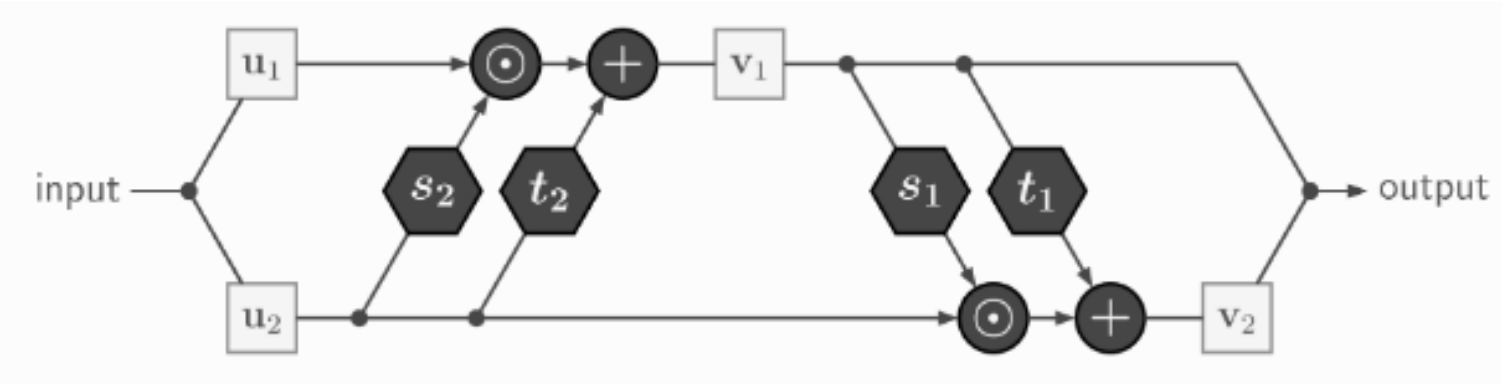
Problem Specification (3)

Finally, the posterior is approximated by conditional transformation of random variable z:

$$q(\mathbf{x} = INN^{-1}(\mathbf{y}, \mathbf{z}; \theta)|\mathbf{y}) = p(\mathbf{z})|J_{\mathbf{x}}|^{-1}$$

$$J_{\mathbf{x}} = det(\frac{\partial INN^{-1}(\mathbf{y}, \mathbf{z}; \theta)}{\partial[\mathbf{y}, \mathbf{z}]}|_{\mathrm{y}, f_{\mathrm{z}}(\mathrm{x})})$$

Architecture (1)

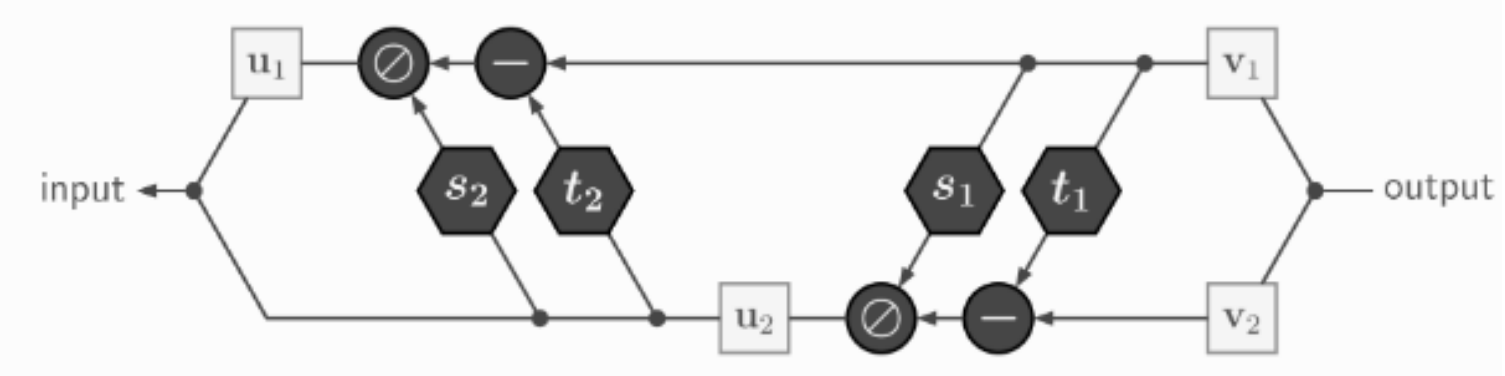


$$\mathbf{v}_1 = \mathbf{u}_1 \odot exp(s_2(\mathbf{u}_2)) + t_2(\mathbf{u}_2)$$
$$\mathbf{v}_2 = \mathbf{u}_2 \odot exp(s_1(\mathbf{v}_1)) + t_1(\mathbf{v}_1)$$

s_i, t_i are arbitrary and potentially non-differentaible functions. Here: neural networks depending on θ.

Architecture (2)

... the operations can be easily reversed:



$$\mathbf{u}_2 = (\mathbf{v}_2 - t_1(\mathbf{v}_1)) \odot exp(-s_1(\mathbf{v}_1))$$
$$\mathbf{u}_1 = (\mathbf{v}_1 - t_2(\mathbf{u}_2)) \odot exp(-s_2(\mathbf{u}_2))$$

Architecture (3)

Short recap::

- input vector **u** is split into two non-overlapping partitions u_1, u_2.
- s, t can be any arbitrarily complicated functions and need not be invertible.
- shuffle the elements of the subsequent layer's input in a randomized way to enhance interaction among the individual variables.
- Pad both the in- and output of the network with an equal number of zeros, if input dimension is small.

Bi-directional Training

Loss $\mathcal{L} = \lambda_y \mathcal{L}_\mathbf{y} + \lambda_z \mathcal{L}_\mathbf{z} + \lambda_x \mathcal{L}_\mathbf{x}$.

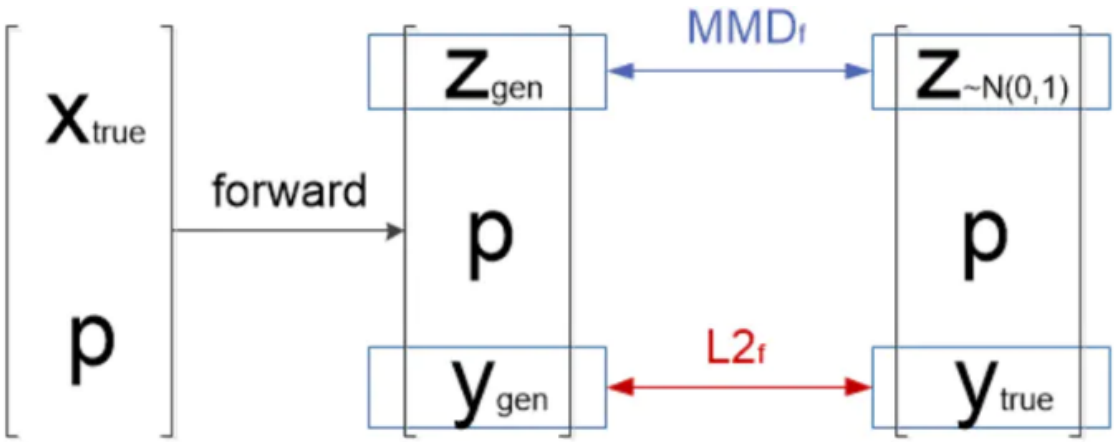
$\mathcal{L}_\mathbf{y}$: forward pass

$\mathcal{L}_\mathbf{x}$: inverse pass

$\mathcal{L}_\mathbf{z}$: independence of z, y

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* $p(\mathbf{x} \mid \mathbf{y}^*)$ *for any measurement* $\mathbf{y}^*$.

Bi-directional Training: Forward pass



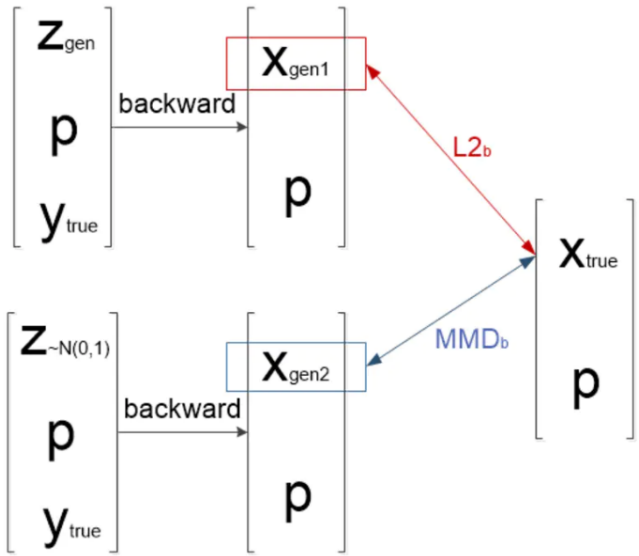
Loss terms involved in **forward process**:

- $\mathcal{L}_{\mathbf{z}}(q(\mathbf{y}, \mathbf{z}), p(\mathbf{y})p(\mathbf{z}))$ (blue arrow)
  Normality of z and independence of y and z are enforced.
- $\mathcal{L}_{\mathbf{y}}(\mathbf{y}, f_{\mathbf{y}}(\mathbf{x}))$ (red arrow)
  Supervised loss for evaluation of our surrogate model $INN(x) = [\hat{y}, \hat{z}]$ vs $[y]$ (e.g. L2 norm)

Bi-directional Training: Inverse pass

$\mathcal{L}_x$ characterizes **backward process** and mainly speeds up convergence:

- reconstruction quality (red arrow), Minimize $||INN^{-1}(y, z) - x||_2^2$
- MMD (blue arrow)
  Similarity of data prior $p(x)$ and marginalized posterior of our INN $q(x) = \int_y q(x|y)$



Comparing probability distributions

We would like to answer whether two probability distributions p and q are equal. We could compute a Z-statistics in case of Gaussians:

$$Z = \frac{\mu_p - \mu_q}{\sqrt{\sigma_p^2/n + \sigma_q^2/n}} \tag{6}$$

Suppose the distribution is non-Gaussian: Wilcoxon signed rank test, Kolmogorov Smirnov test, ...

Maximum mean discrepancy (MMD)[**?**] is another test for comparing potentially non-Gaussian distributions. It can be seen as a **kernel two-sample test**.

Maximum Mean Discrepancy

Main assumption of MMD:

$$E_{x\sim p}(f(x)) = E_{y\sim q}(f(y)) \text{ if and only if } p = q$$

for any **well-behaved function** (bounded) function $f \in \mathcal{F}(\mathcal{X})$ on some space $\mathcal{X}$.

MMD quantifies that difference of expectations by maximizing wrt. f as of

$$MMD(p, q) = \sup_{f\in\mathcal{F}(\mathcal{X})} [E_{x\sim p}(f(x)) - E_{y\sim q}(f(y))]$$

Example

Let's try certain naive feature mappings $f(x) = \langle f, \varphi(x) \rangle$.
$\varphi(x) = x$ allows us to focus on the first moment, i.e.

$$\begin{aligned} MMD(p,q) &\propto ||E_{x\sim p}(\varphi(x)) - E_{y\sim q}(\varphi(y))|| \\ &= ||E_{x\sim p}(x) - E_{y\sim q}(y)|| \\ &= ||\mu_p - \mu_q|| \end{aligned}$$

Let's add the 2nd moment: $\varphi(x) = (x, x^2)$:

$$\begin{aligned} MMD(p,q) &\propto ||E_{x\sim p}(\varphi(x)) - E_{y\sim q}(\varphi(y))|| \\ &= \left|\left|[E_{x\sim p}(x), E_{x\sim p}(x^2)] - [E_{y\sim q}(y), E_{y\sim q}(y^2)]\right|\right| \\ &= \sqrt{(E_{x\sim p}(x) - E_{y\sim q}(y))^2 + (E_{x\sim p}(x^2) - E_{y\sim q}(y^2))^2} \end{aligned}$$

Kernel trick

We can compare more complex moments by restricting $\mathcal{F}(\mathcal{X})$ as unit ball in a reproducing kernel Hilbert space and defining a feature mapping $k(x, y) = \langle \varphi(x), \varphi(y) \rangle$. $MMD^2(p, q)$ now becomes:

$$
\begin{aligned}
MMD^2(p, q) &= \max_{f:||f|| \leq 1} \left[E_{x \sim p}(f(x)) - E_{y \sim q}(f(y)) \right]^2 \\
&= \dots \\
&\propto ||\mu_p - \mu_q||^2 \\
&= \dots \\
&= E_{x,x'} \left\langle \varphi(x), \varphi(x') \right\rangle + E_{y,y'} \left\langle \varphi(y), \varphi(y') \right\rangle - 2E_{x,y} \left\langle \varphi(x), \varphi(
\\
&= E_{x,x'} k(x, x') + E_{y,y'} k(y, y') - 2E_{x,y} k(x, y)
\end{aligned}
$$

meaning that MMD quantifies inter-distribution similarity $k(x, x'), k(y, y')$ as well as cross-distribution similarity $k(x, y)$.

Kernel trick

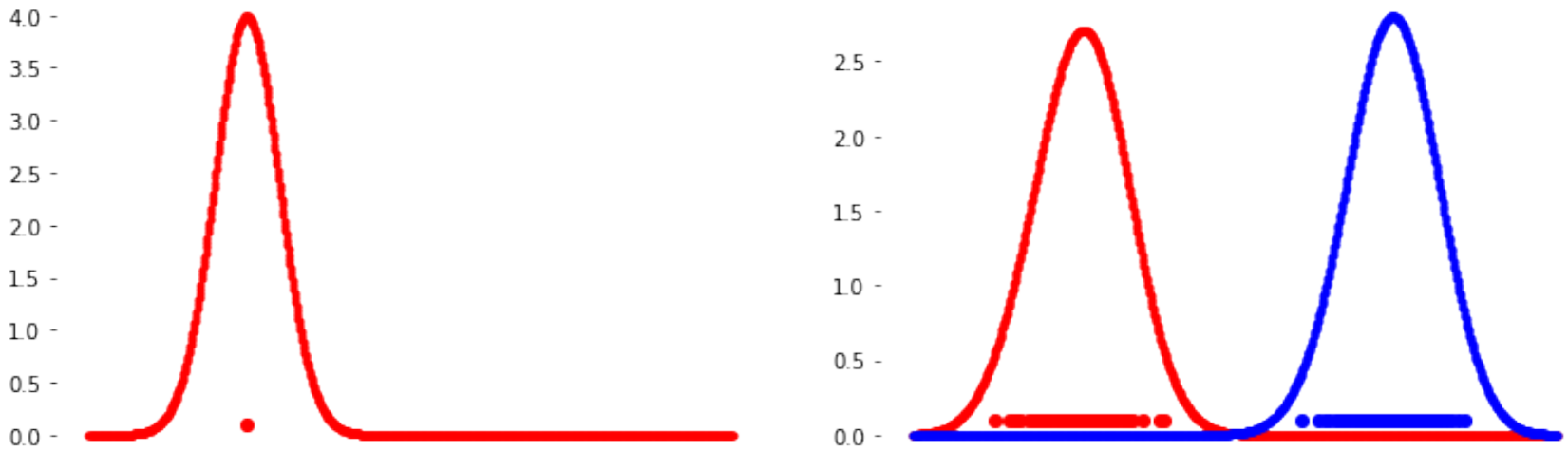
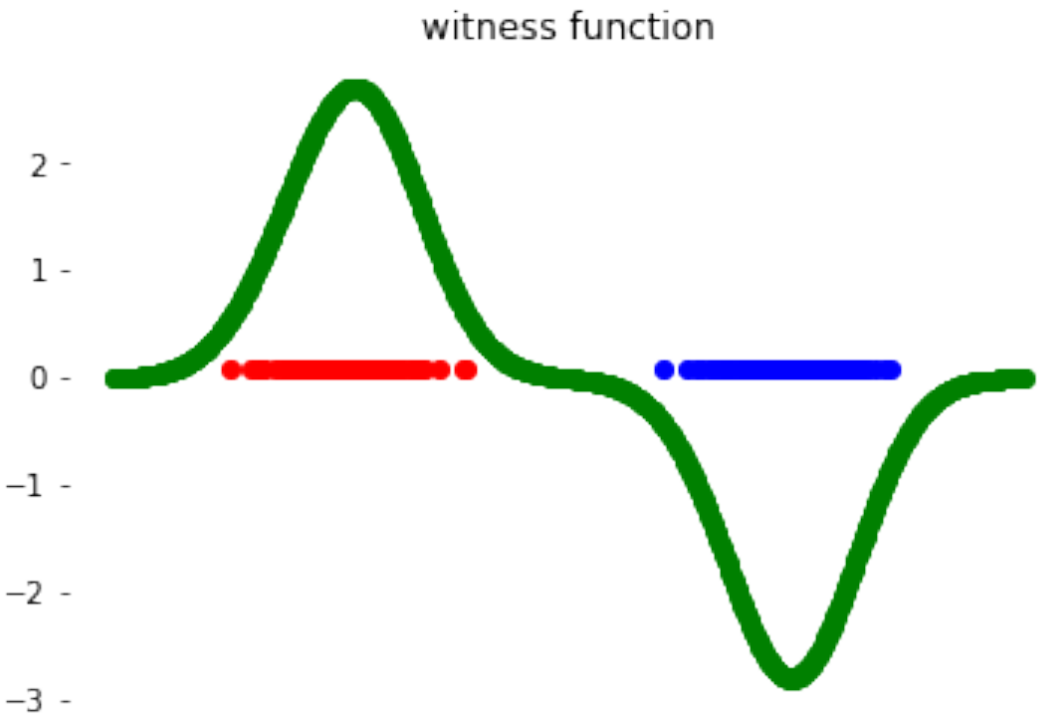


Figure: Our distributions p (red) and q (blue) are interpolated in terms of a certain kernel (here: Gaussian) at any $x_i \sim p$, $y_i \sim q$.

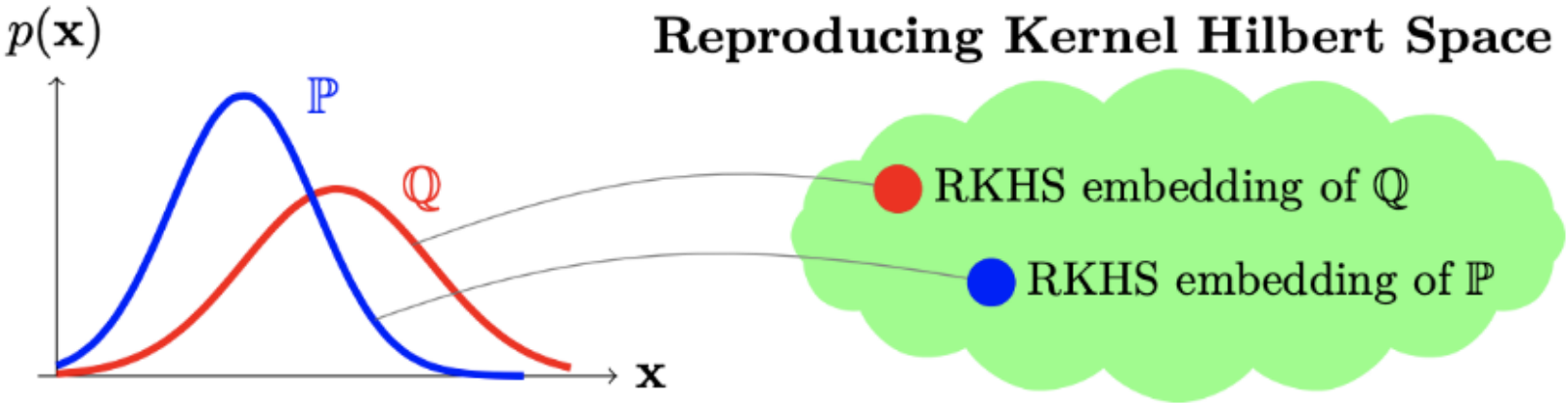
Witness function

MMD is basically comparing p and q by witness function $w(p, q)$, i.e.

$$MMD^2(p, q) = ||\mu_p - \mu_q||^2 = ||w(p, q)||^2$$



MMD loss of invertible networks



MMD is easier to use, more stable and cheaper to compute than loss terms based on discriminator of GANs (Tolstikinhin et al., 2017).

Inverse multiquadratics kernel found by optimization:
$k(\mathbf{x}, \mathbf{x}') = 1/(1 + \left\|(\mathbf{x} - \mathbf{x}')/h\right\|_2^2)$, where h defines the width of the kernel.