Invertible Neural Networks as a tool for ill-posed inverse problems

Jakob Kruse

16/09/2019

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Visual Learning Lab Heidelberg

European Research Council
Established by the European Commission

Introduction

Lynton Ardizzone

Jakob Kruse

Ullrich Köthe

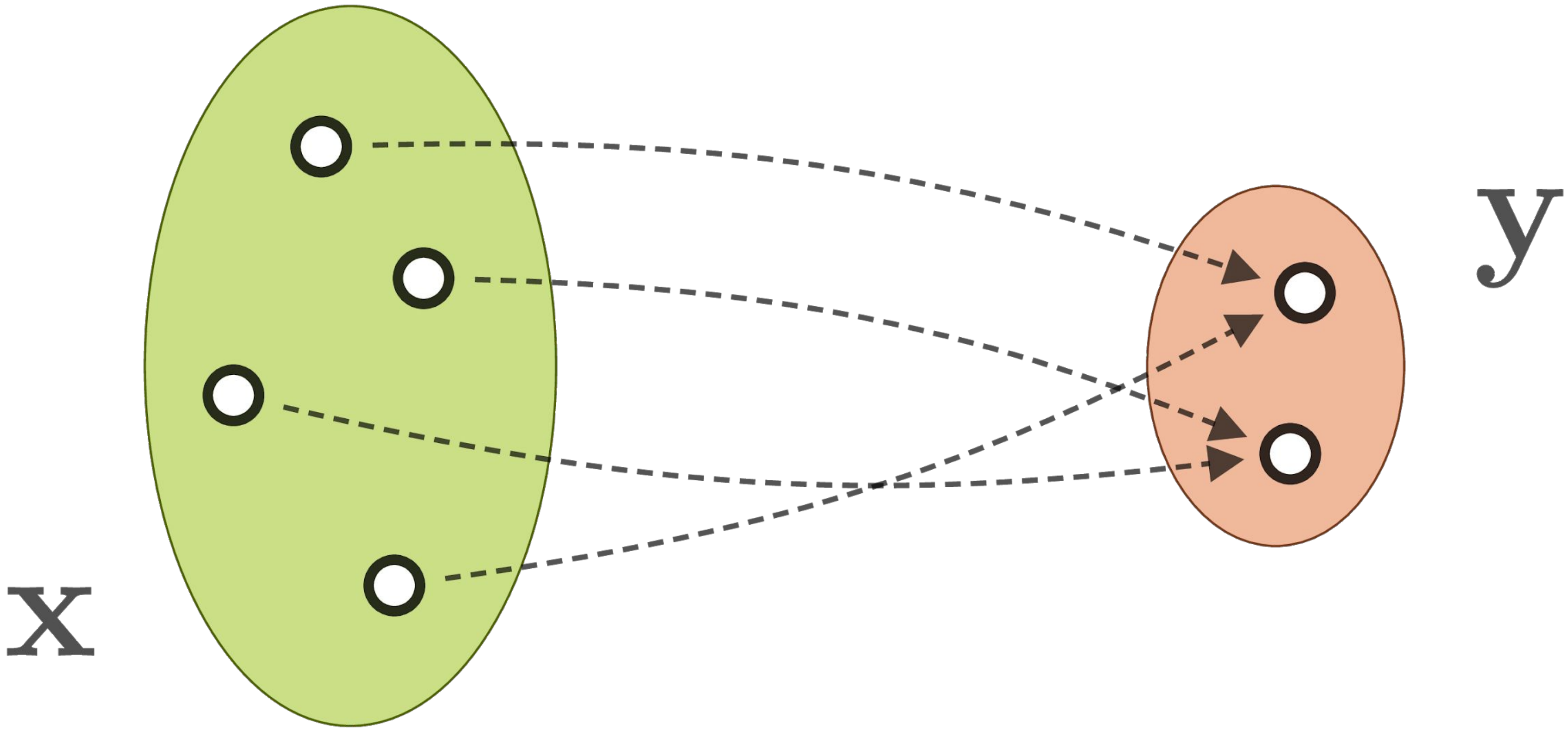
Carsten Rother

Based in Heidelberg

3 PIs, ~15 PhD students/postdocs

Computer Vision, Graphical Models, Invertible Networks, Uncertainty, Causality, Explainable Machine Learning

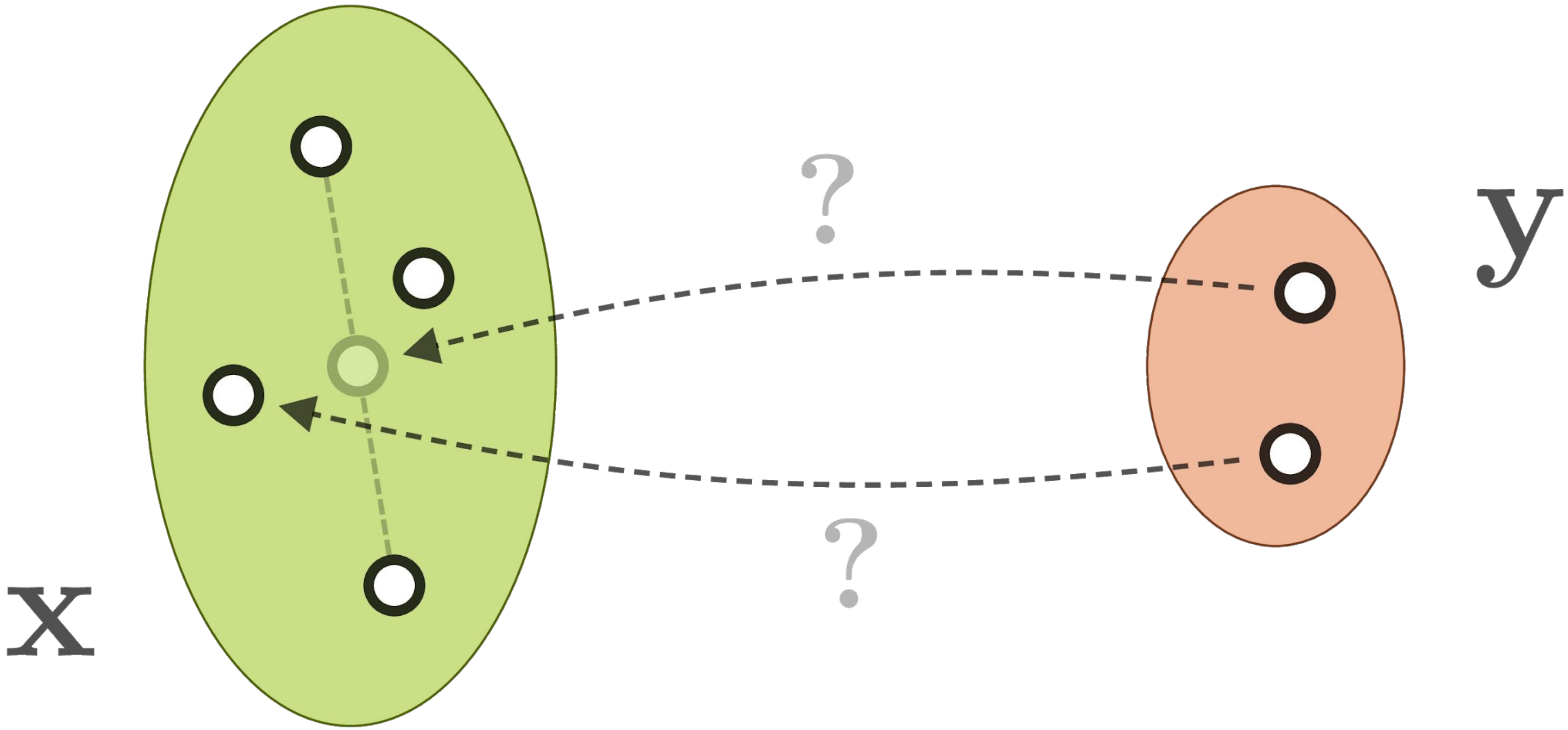
What are inverse problems?



Forward process:

Many parameters **x** may map to same observation **y**

What are inverse problems?

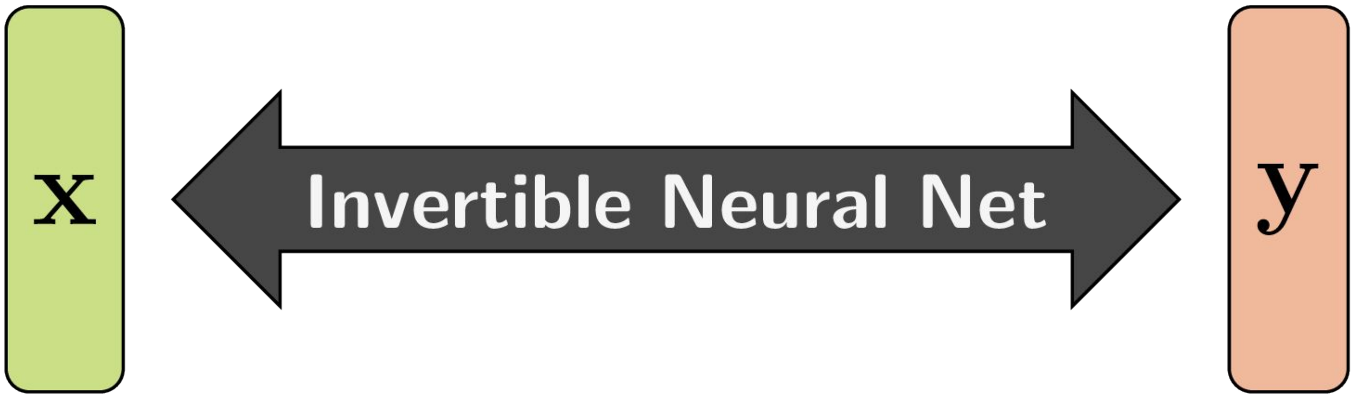


The *inverse problem* is ambiguous and ill-posed

Regularization can guide towards most likely solution

But actually we want conditional probabilities

What are Invertible Neural Networks?



Bijective mapping between two domains

Both directions efficient to compute

Tractable Jacobian determinant

Input and output must have equal dimensions

Invertible Neural Networks: i-ResNet [6]

Standard ResNet block, but with contractive residual

$$\mathbf{x}_{t+1} = \mathbf{x}_t + g_{\theta_t}(\mathbf{x}_t)$$

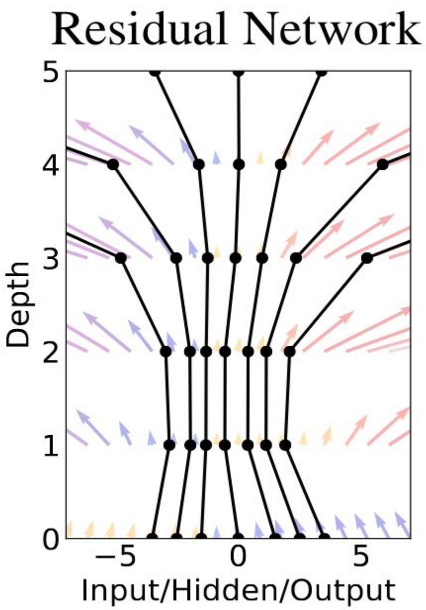
Inverse via fixed-point iteration:

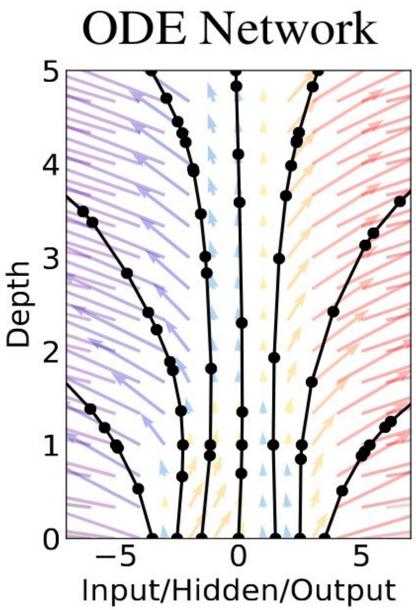
$$\mathbf{x}_t^0 = \mathbf{x}_{t+1}$$
$$\mathbf{x}_t^{i+1} = \mathbf{x}_{t+1} - g_{\theta_t}(\mathbf{x}_t^i)$$

Approximate Jacobian determinant via power series:

$$\ln\left|\det\left(I + J_{g_{\theta_t}}(\mathbf{x}_t)\right)\right| \approx \sum_{k=1}^{n} (-1)^{k+1} \frac{\mathrm{tr}\left(J_{g_{\theta_t}}(\mathbf{x}_t)^k\right)}{k}$$

From discrete, block-wise transformations to continuous flow field traversed via ODE solver





$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Invertible Neural Networks: Various other options

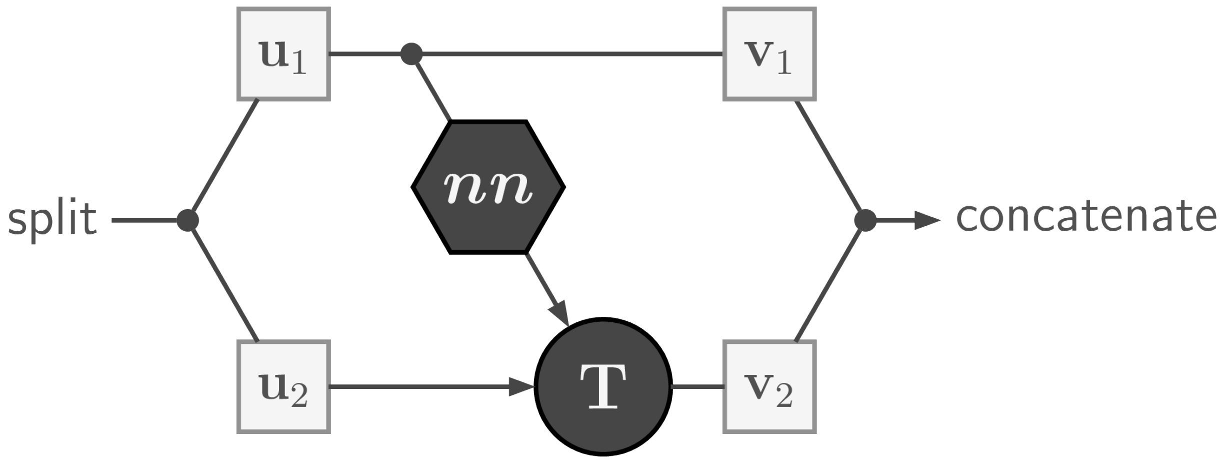
Orthogonal weight matrices [8]

Invertible convolutions in Fourier space [9]

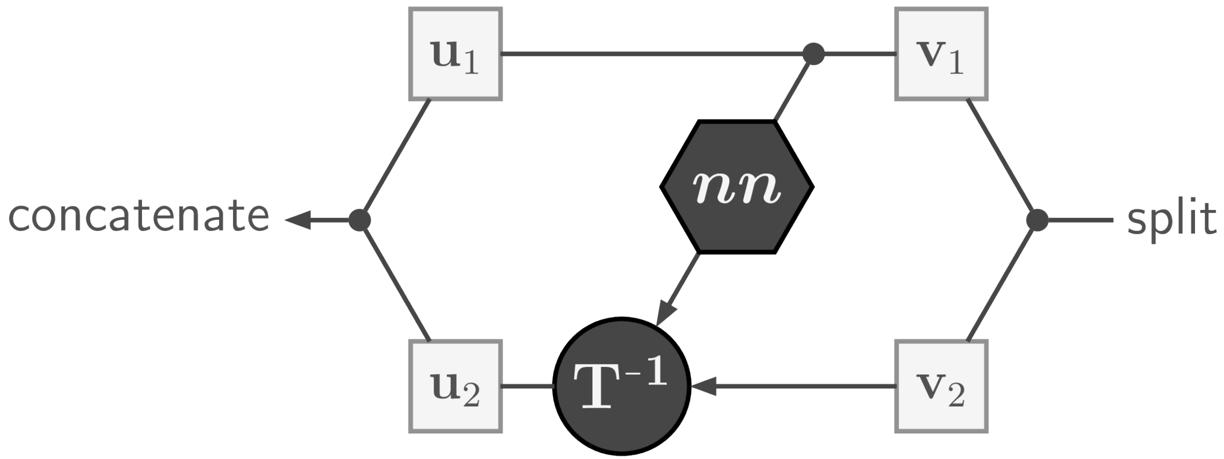
Autoregressive models: asymmetric cost

Can be alleviated [10]

Autoencoder, VAE: no Jacobian determinant

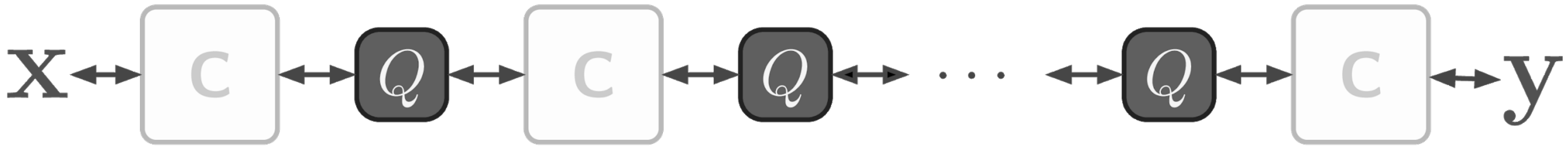


$\mathbf{u}_1$
$\mathbf{v}_1$
nn
split
concatenate
$\mathbf{u}_2$
T
$\mathbf{v}_2$



$\mathbf{u}_1$
$\mathbf{v}_1$
nn
concatenate
split
$\mathbf{u}_2$
$\mathbf{T}^{-1}$
$\mathbf{v}_2$

Chain coupling blocks (**C**) together like residual blocks

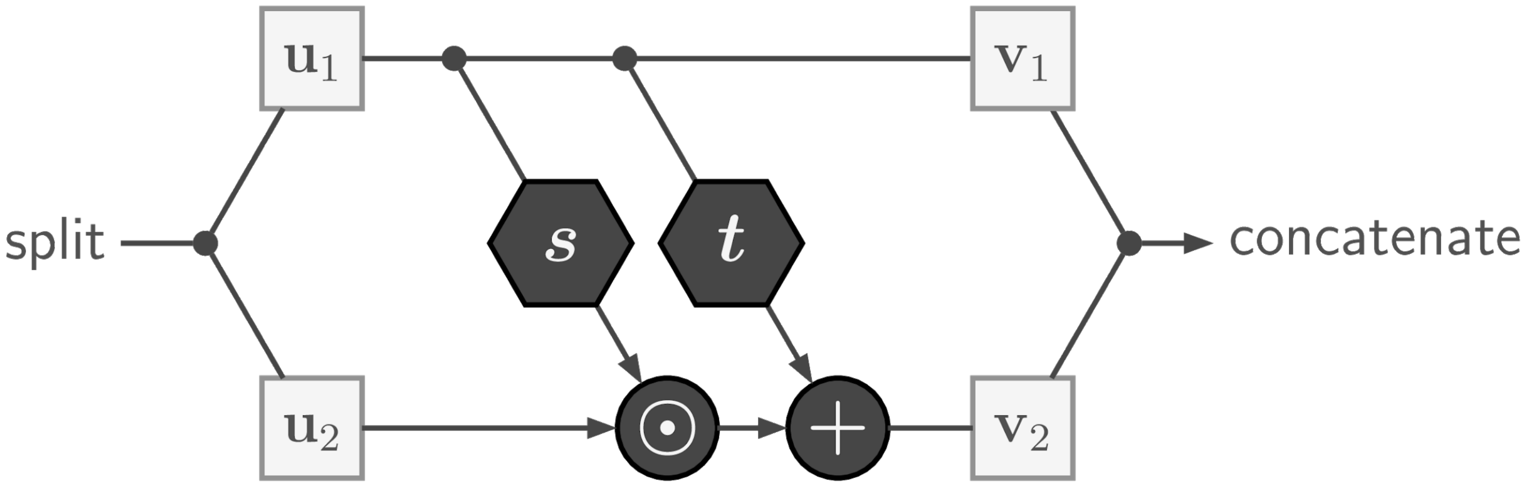


Important: permute variables between blocks!

Permutations Q are easy to invert

Can be relaxed to (learned) orthogonal matrices

Split and permute *channels* for images/embeddings

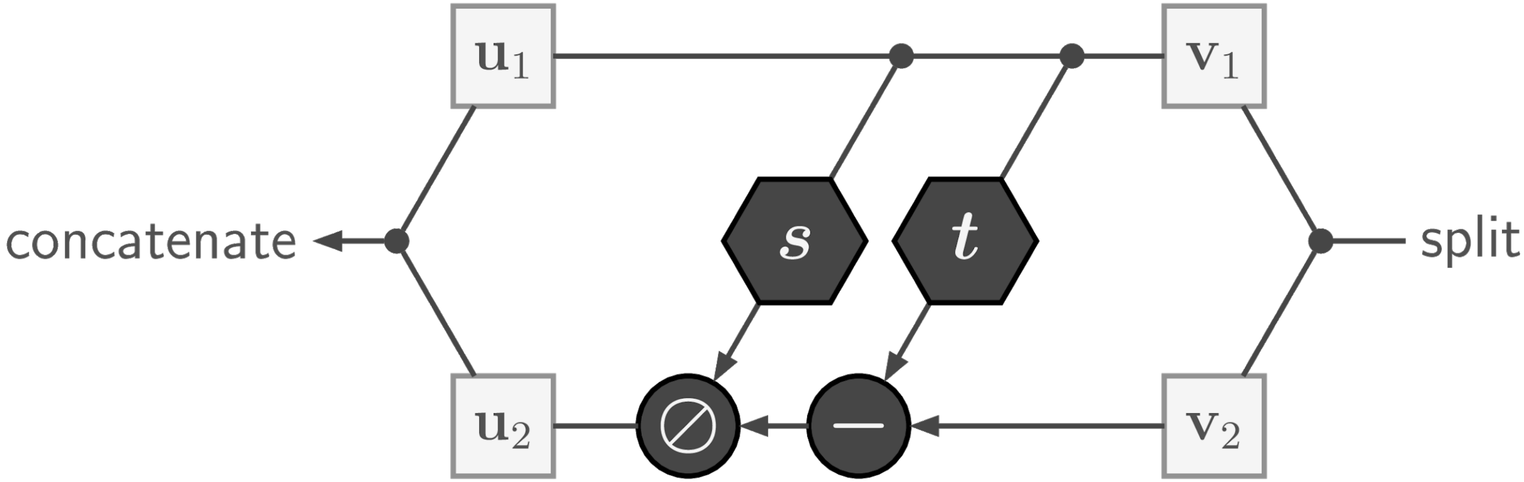


$$\mathbf{v}_1 = \mathbf{u}_1$$
$$\mathbf{v}_2 = \mathbf{u}_2 \odot \exp\big(s(\mathbf{u}_1)\big) + t(\mathbf{u}_1)$$

Following “Real NVP” architecture [4]:

Transformation **T** consists of scaling **s** and translation **t**

Invertible Neural Networks: Affine Coupling Blocks



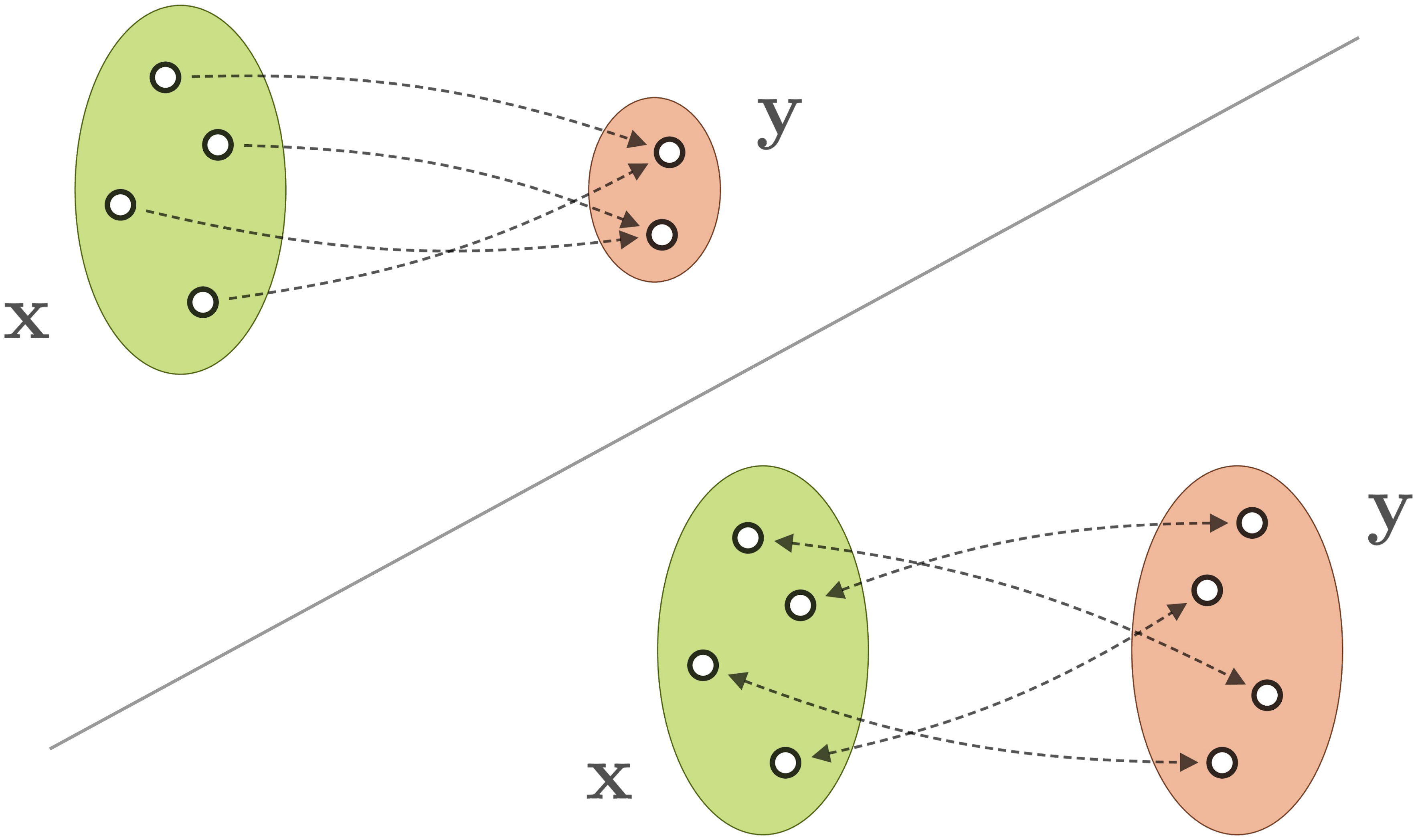
$$\mathbf{u}_1 = \mathbf{v}_1$$

$$\mathbf{u}_2 = \big(\mathbf{v}_2 - t(\mathbf{u}_1)\big) \oslash \exp\big(s(\mathbf{u}1)\big)$$

Following “Real NVP” architecture [4]:

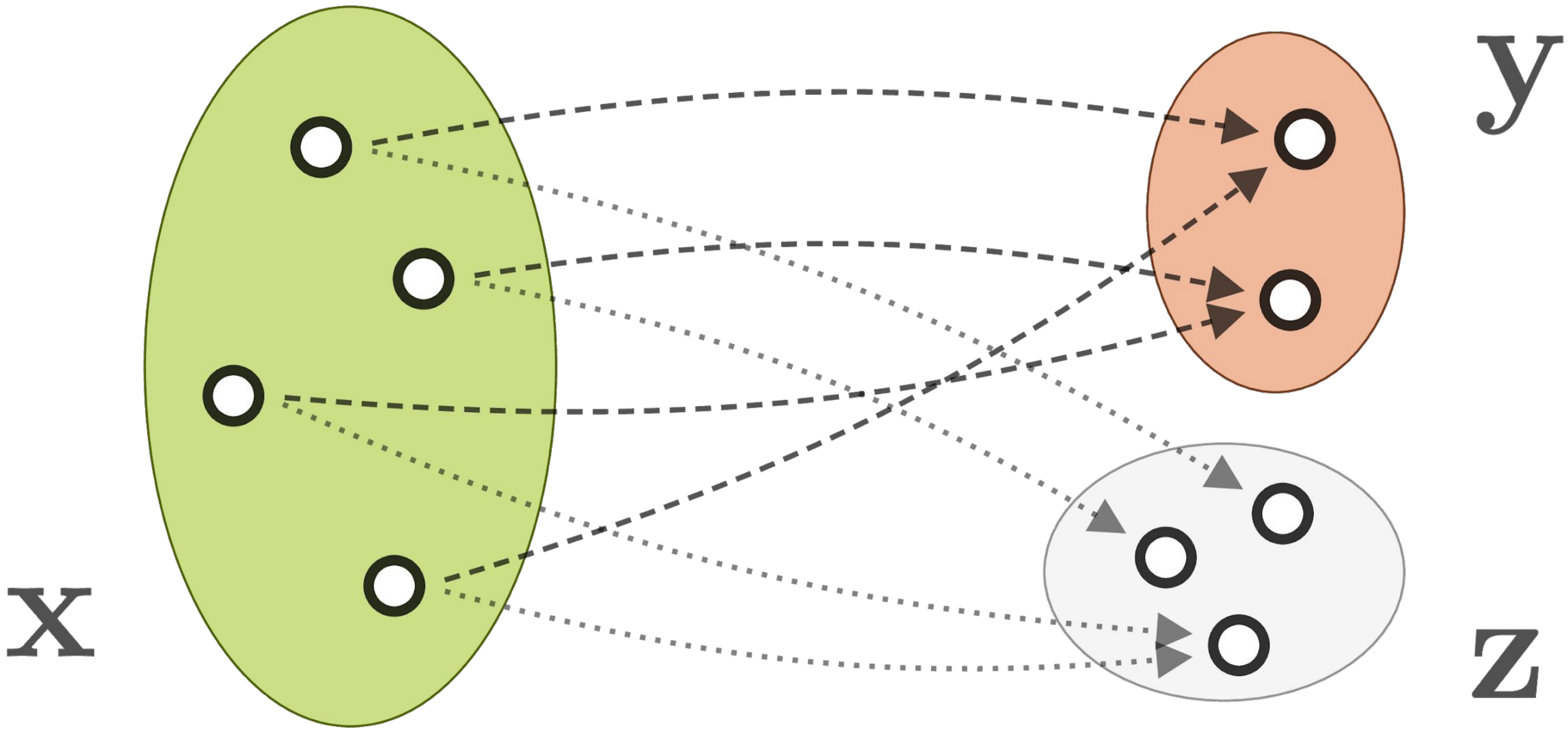
Transformation **T** consists of scaling **s** and translation **t**

Problem: many-to-one mapping is not bijective



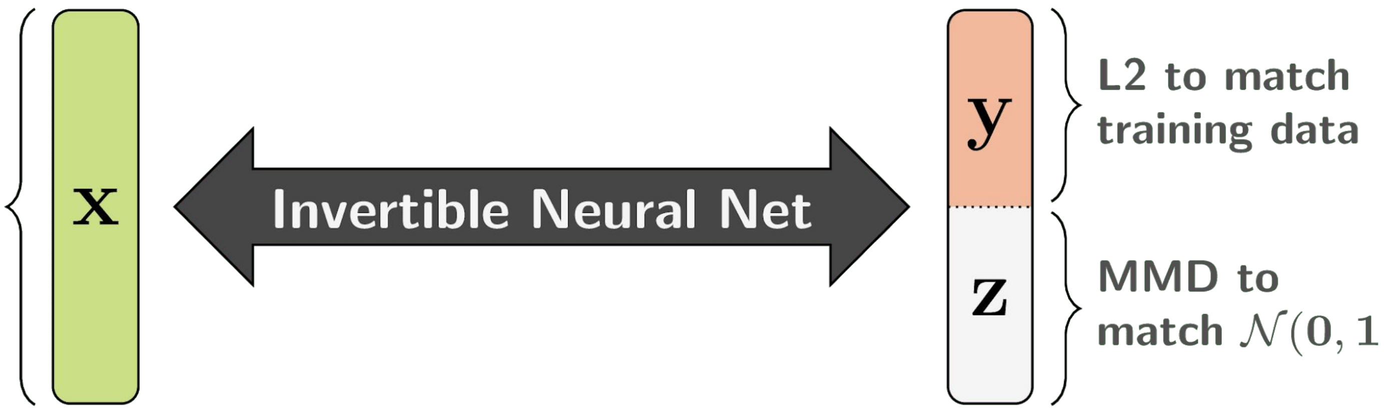
Visual Learning Lab Heidelberg

Fix: augment observations with latent variables



$\mathbf{x} \leftrightarrow [\mathbf{y}, \mathbf{z}]$ is a one-to-one mapping, equal dimensions

Learning augmented mapping with an INN [1]



$$\mathrm{MMD}(\mathbf{u}, \mathbf{u}') = \mathbf{E}_{i,j}[\kappa(\mathbf{u}_i, \mathbf{u}_j)] - 2 \cdot \mathbf{E}_{i,j}[\kappa(\mathbf{u}_i, \mathbf{u}'_j)] + \mathbf{E}_{i,j}[\kappa(\mathbf{u}'_i, \mathbf{u}'_j)] \quad [5]$$

with multiquadratic kernel $\kappa(\mathbf{u}, \mathbf{u}') = \left(1 + \left\| \frac{\mathbf{u}-\mathbf{u}'}{h} \right\|_2^2 \right)^{-1}$

Jointly learn forward process and encoding of the lost information, get the inverse for free!

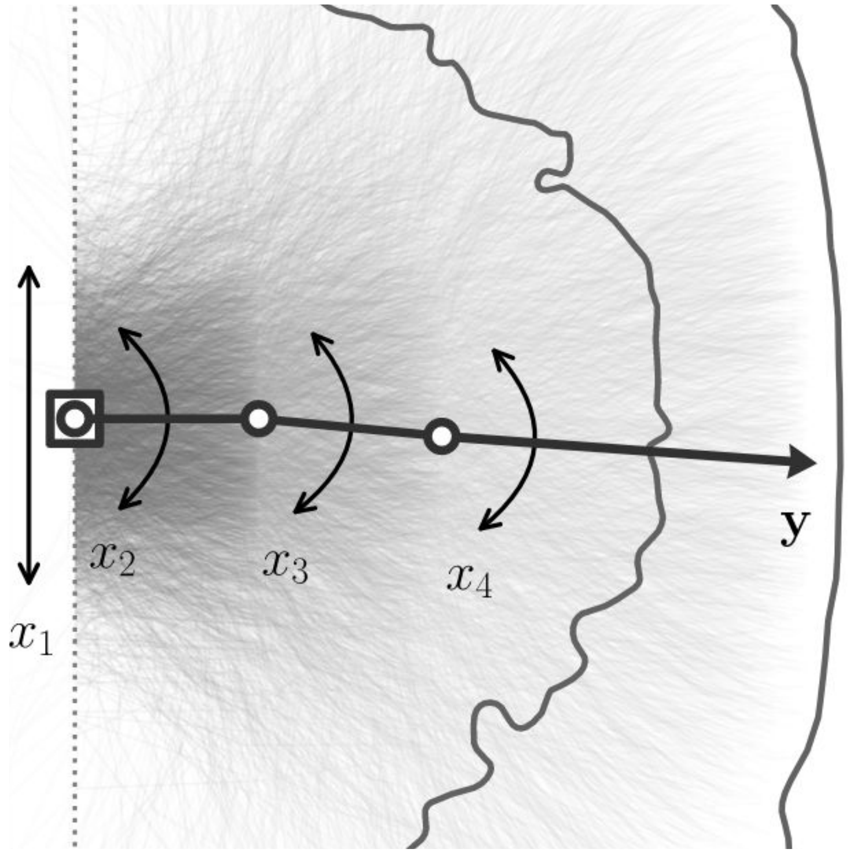
Learning augmented mapping with an INN [1]

To sample p(**x**|**y**), fix **y** and run **z** samples through the net
→ correct posterior if all losses perfectly converged

Data set must represent true **x** distribution

Training must ensure **y** and **z** are independent

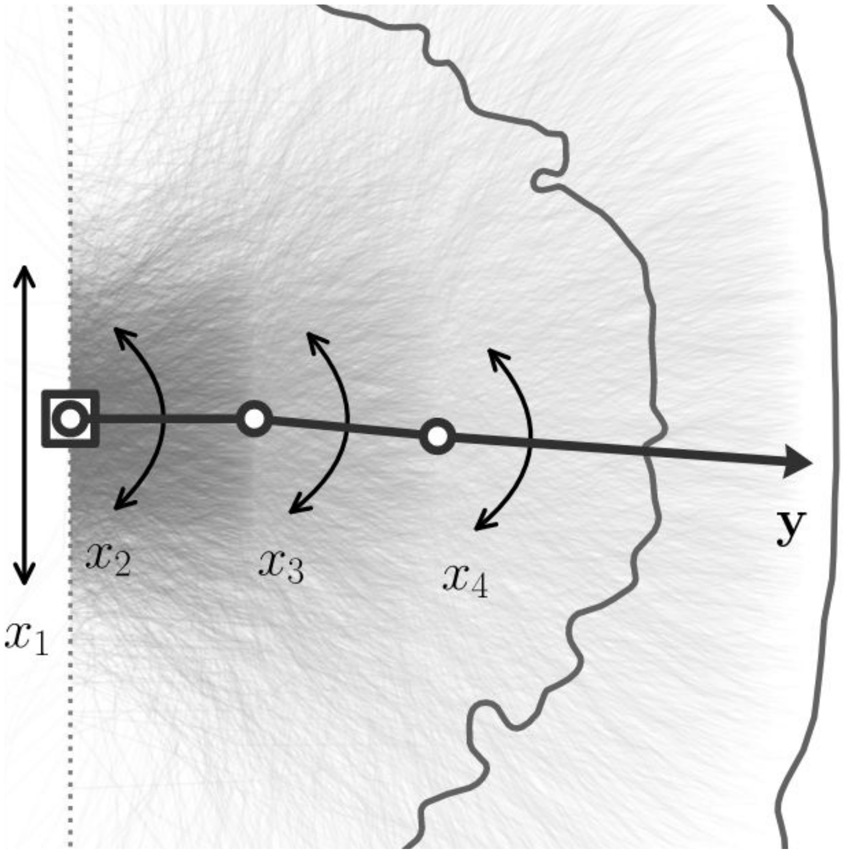
Toy example: Inverse kinematics



2D arm with four degrees of freedom and prior p(**x**)

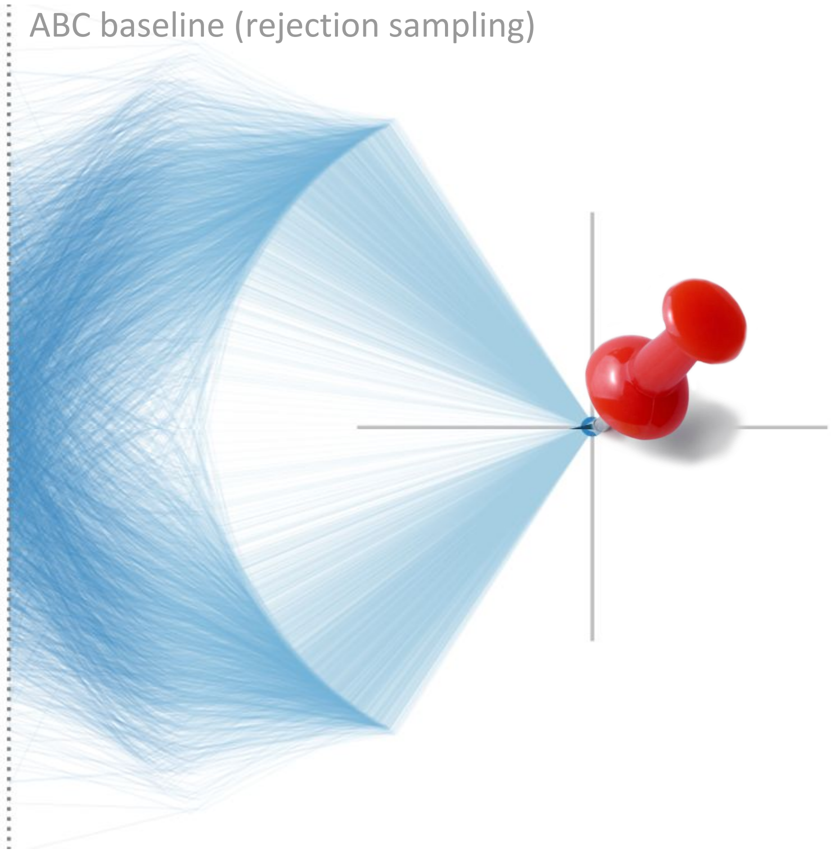
x is the pose, **y** is the end point

Toy example: Inverse kinematics



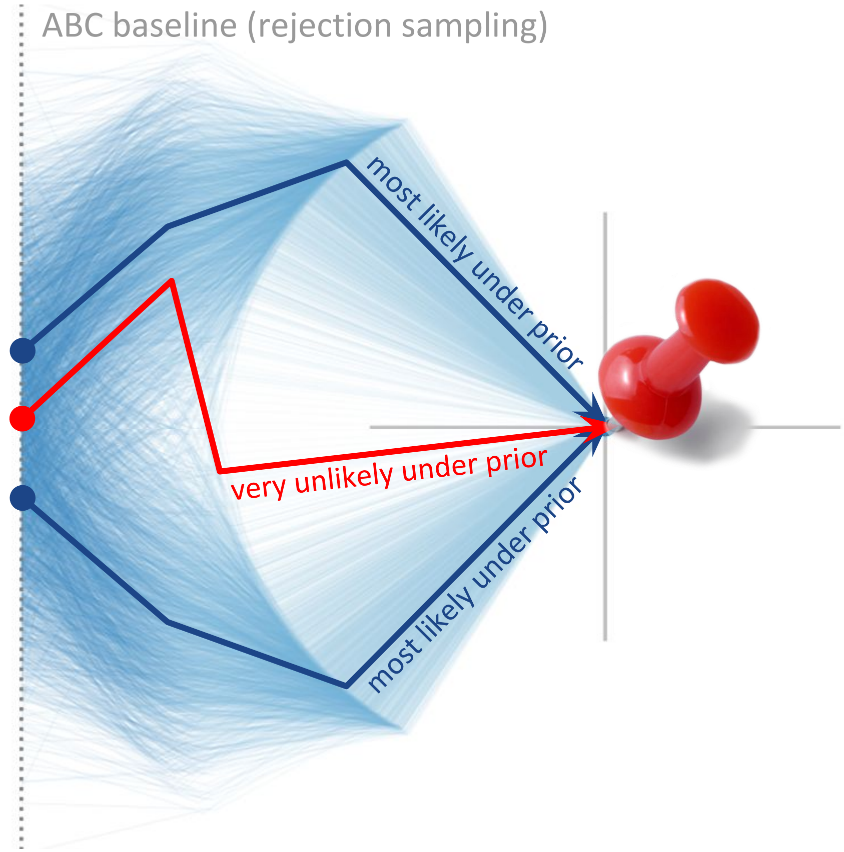
Complex enough to not be trivial, but still allow ground truth and easy visualization

Toy example: Inverse kinematics



Inverse Problem:
Distribution over all poses ending in a given point

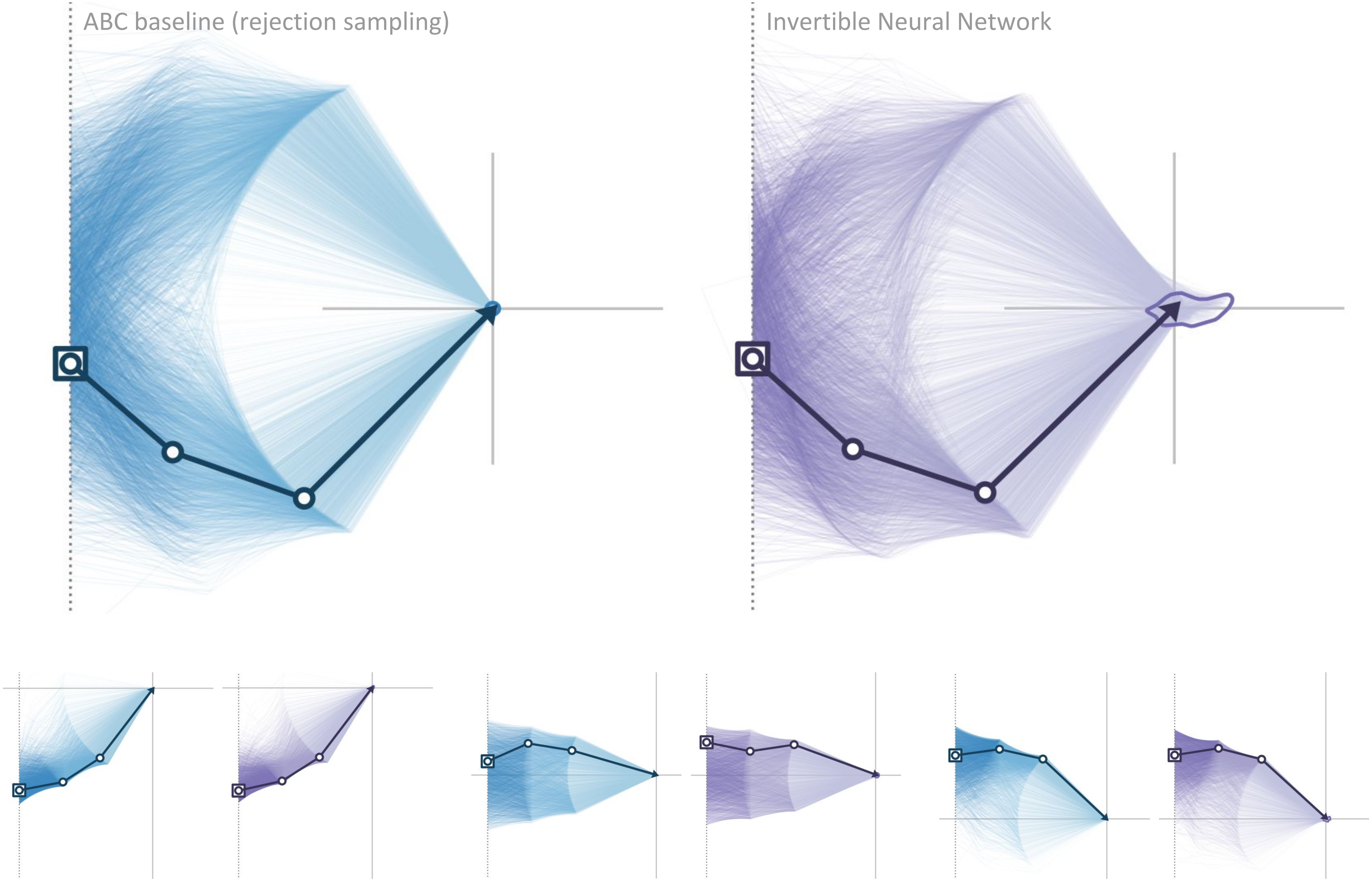
Toy example: Inverse kinematics



Inverse Problem:

Distribution over all poses ending in a given point

Toy example: Inverse kinematics



Real example: Multispectral tissue imaging

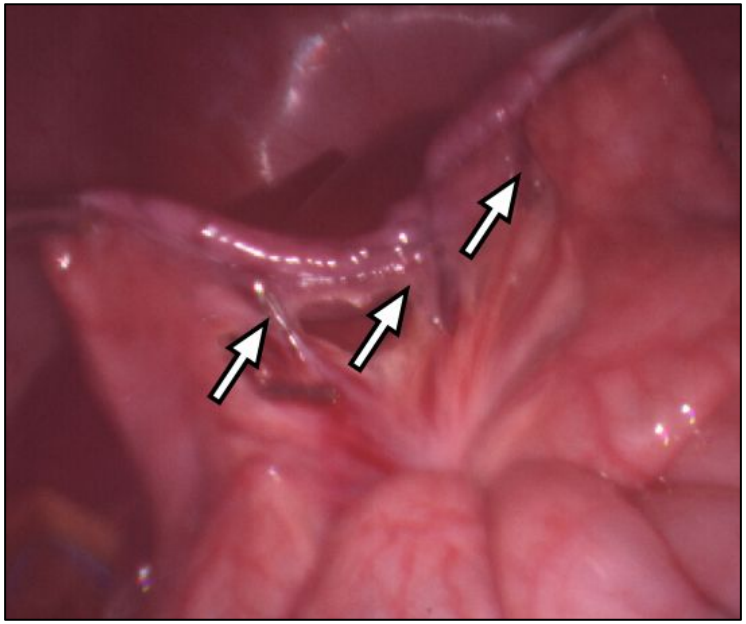
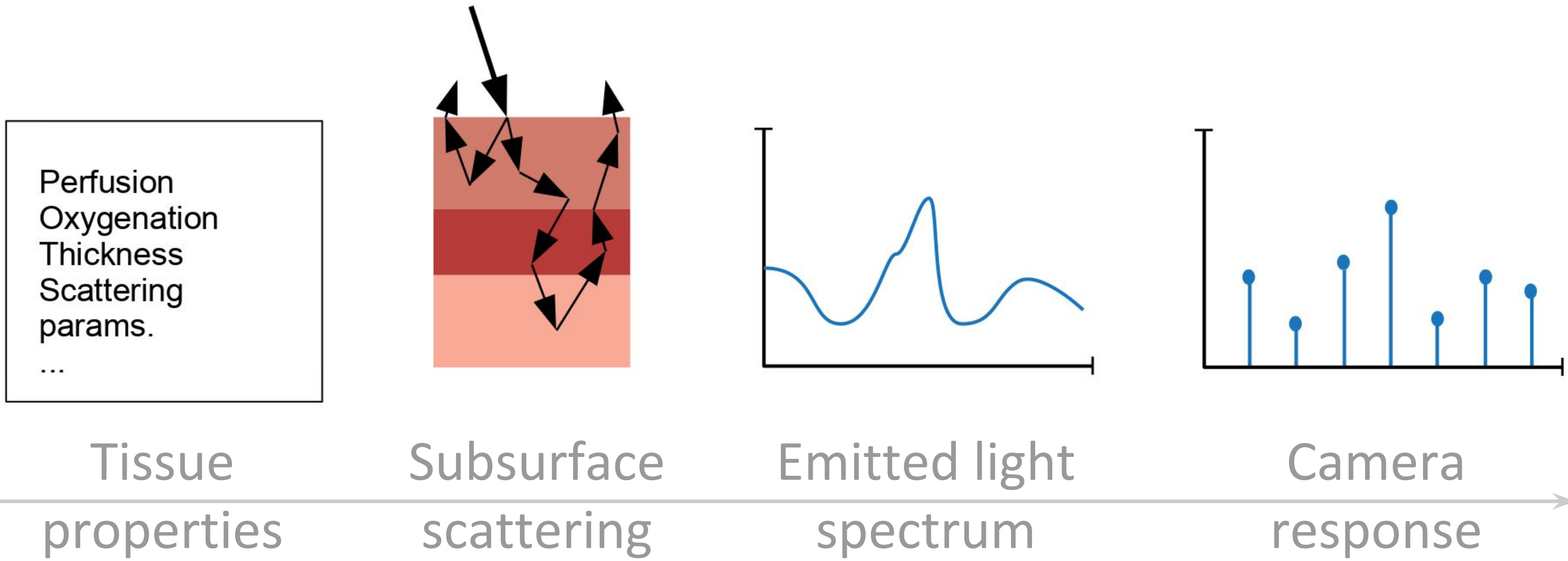


Image from DKFZ Heidelberg, arrows indicate clips

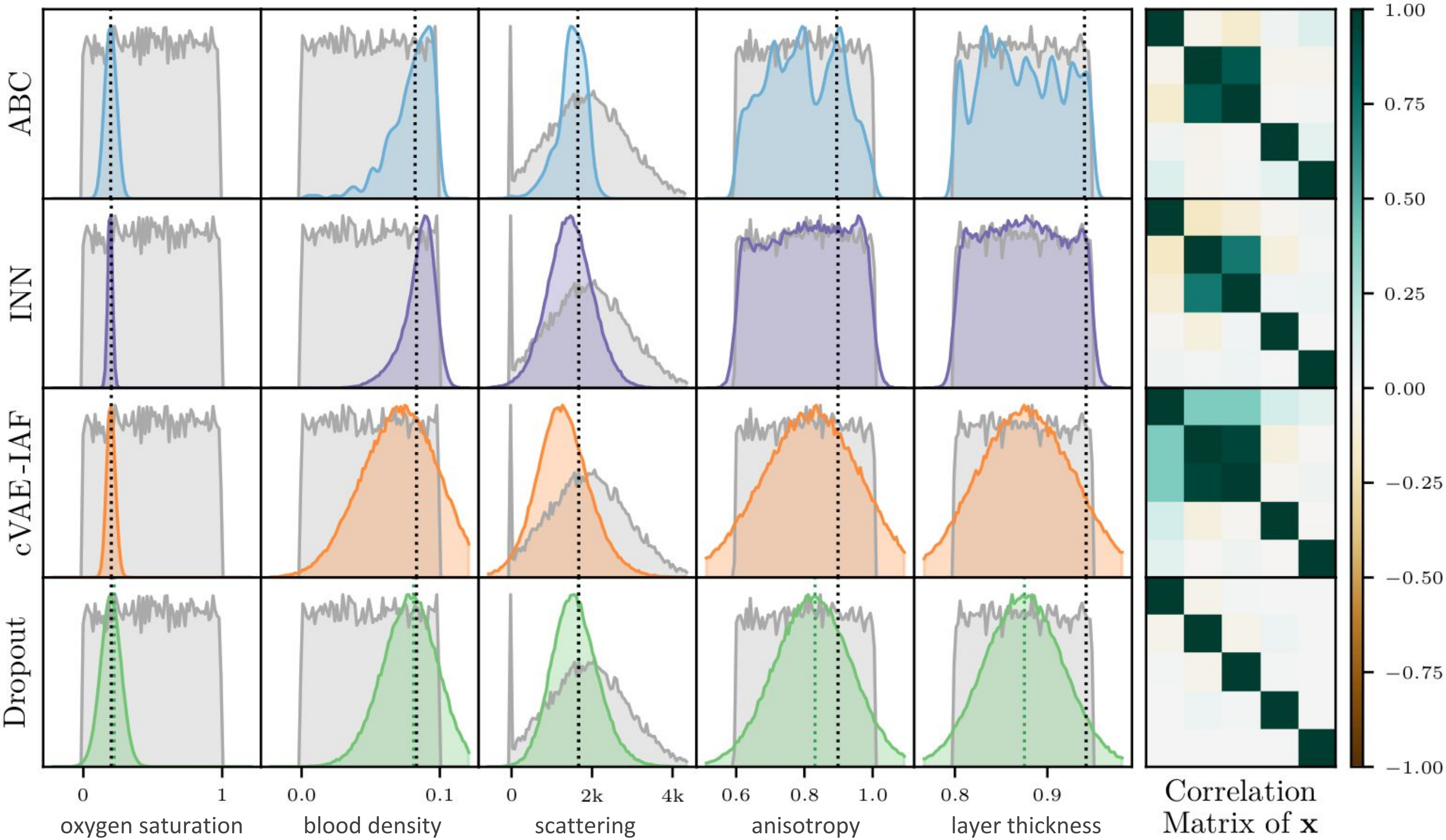
Real example: Multispectral tissue imaging

(Simulated) forward process:

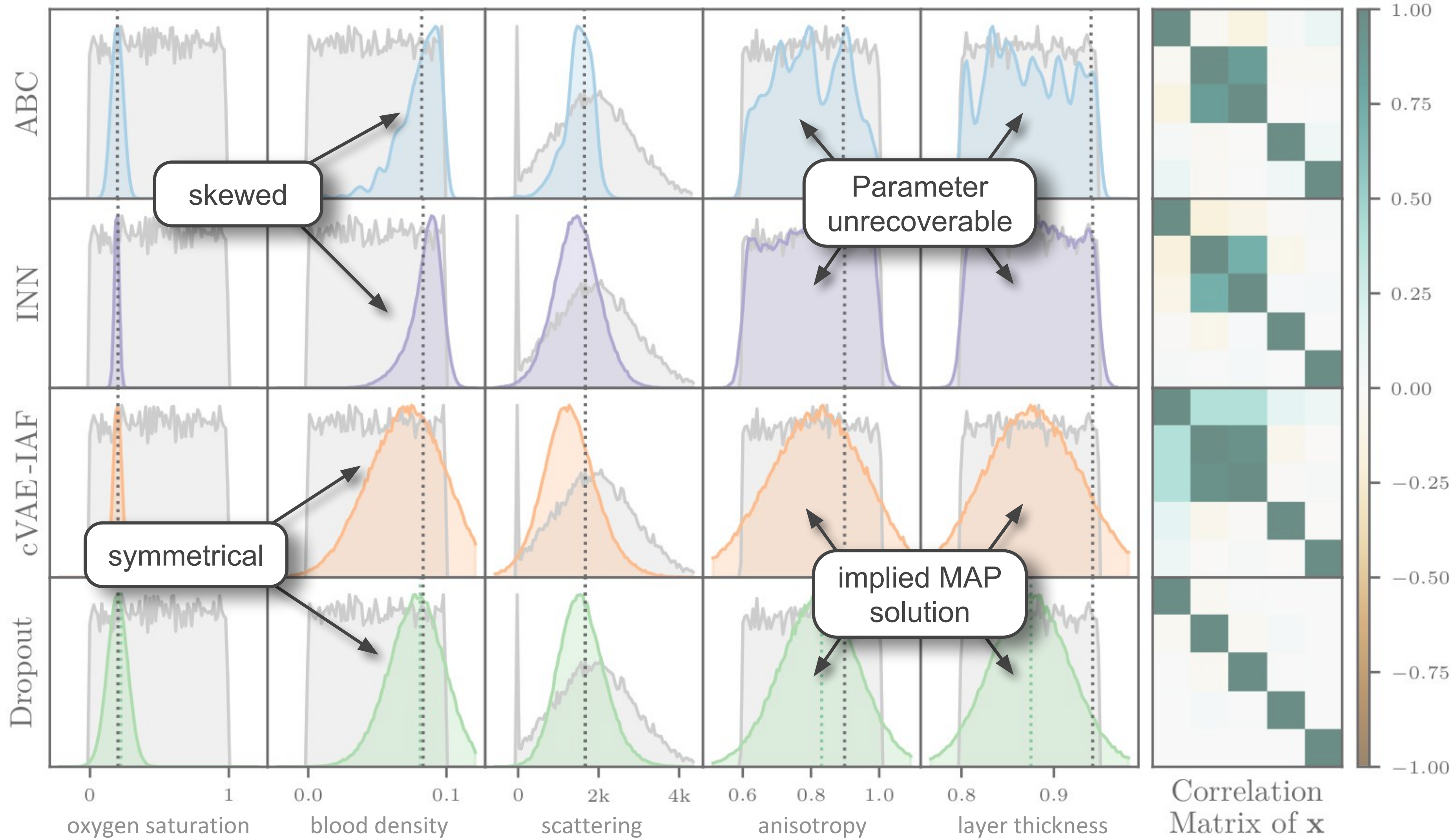


Inverse Problem: Distribution over tissue properties given the multispectral measurement

Real example: Multispectral tissue imaging

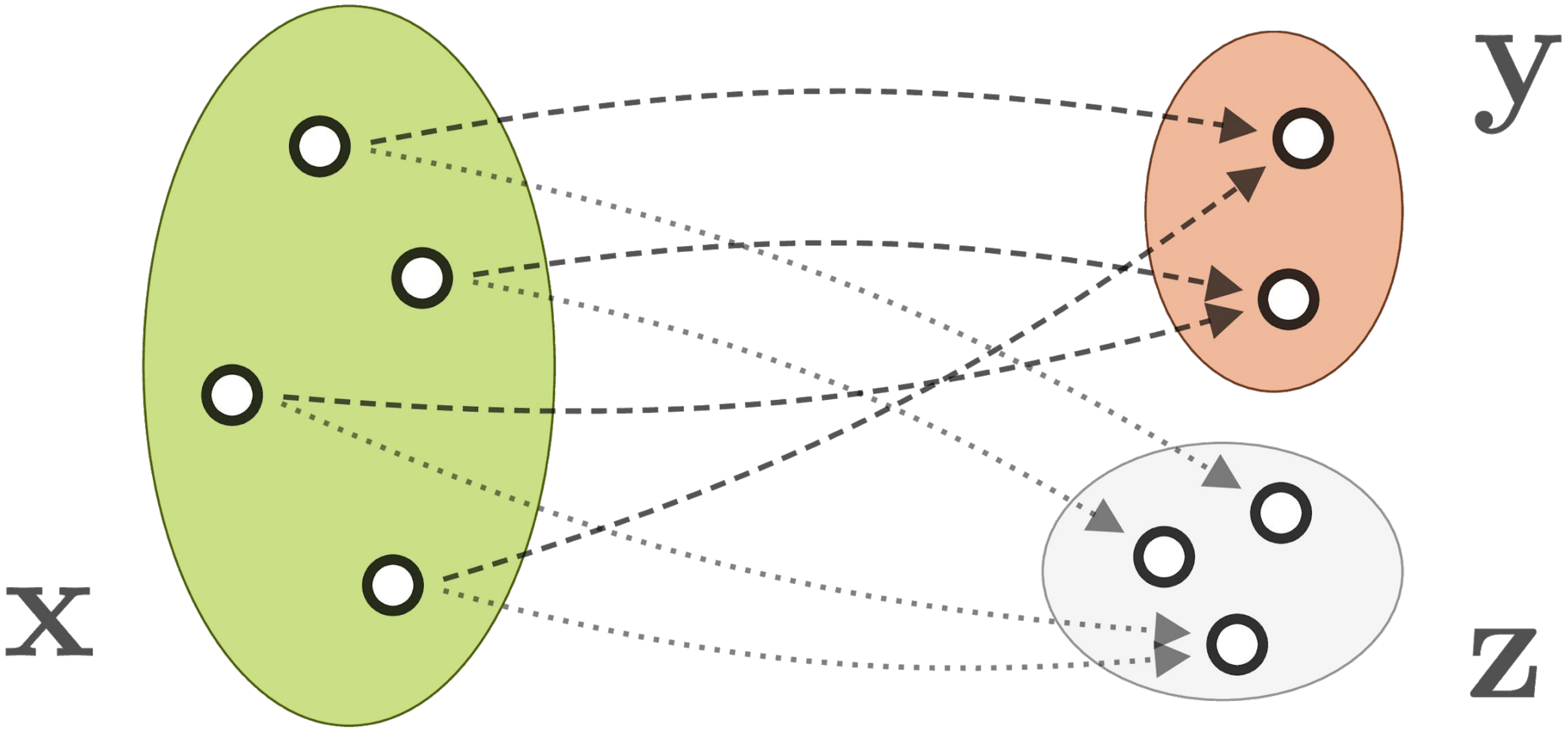


Real example: Multispectral tissue imaging



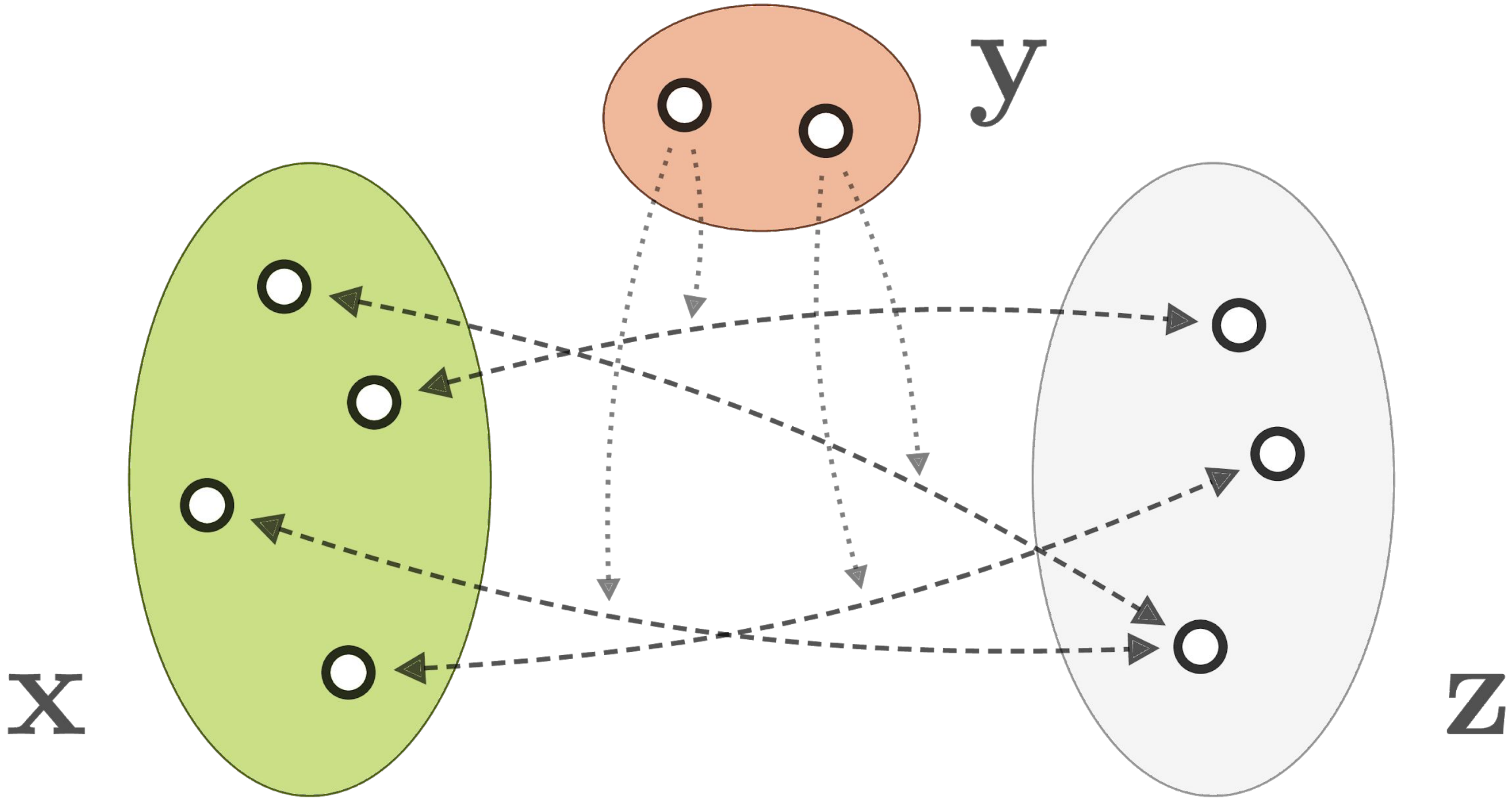
A different perspective

So far, we augmented **y** to make bijection fit



Problems: dimensions need to add up, forward process can't be ambiguous, MMD doesn't scale very well

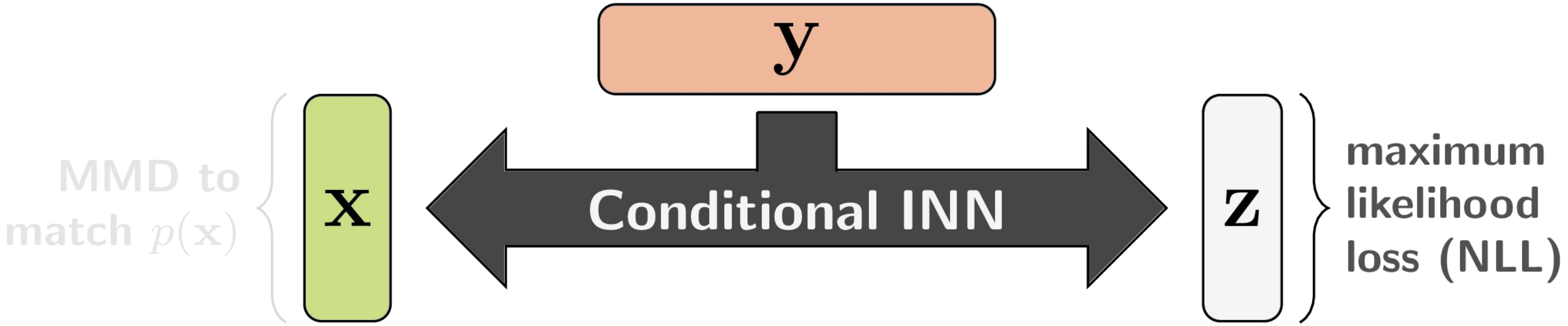
A different perspective



Learn relation between **x** and **z** *conditioned* on **y**

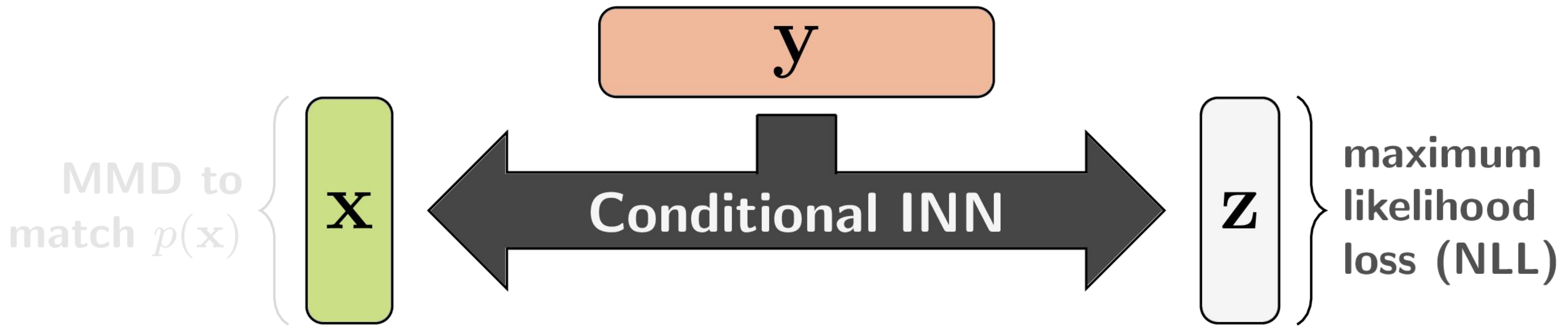
x ↔ **z** is a one-to-one mapping with equal dimensions

Can be learned with conditional INN [3]



$$q_\theta(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{z}) \cdot \left|\det J_{\mathbf{x}|\mathbf{y} \mapsto \mathbf{z}}\right|, \quad p(\mathbf{z}) = \mathcal{N}(0,1)$$

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{y}) &= -\log\left(q_\theta(\mathbf{x} \mid \mathbf{y})\right) \\ &= \frac{1}{2} \cdot \mathbf{z}^2 - \log\left|\det J_{\mathbf{x}|\mathbf{y} \mapsto \mathbf{z}}\right| \end{aligned}$$

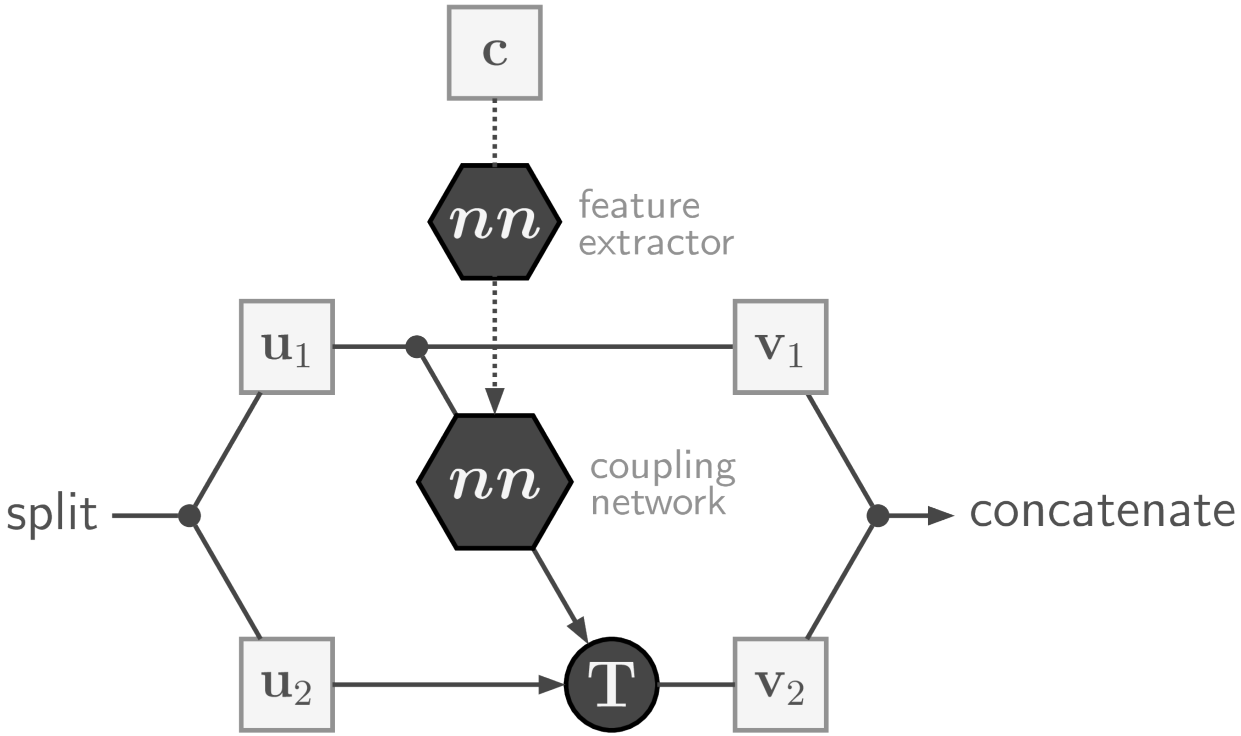


$$q_\theta(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{z}) \cdot \left|\det J_{\mathbf{x}|\mathbf{y} \mapsto \mathbf{z}}\right|, \qquad p(\mathbf{z}) = \mathcal{N}(0,1)$$

$$\begin{aligned}\mathcal{L}(\mathbf{x},\mathbf{y}) &= -\log\left(q_\theta(\mathbf{x} \mid \mathbf{y})\right) \\ &= \frac{1}{2} \cdot \mathbf{z}^2 - \log\left|\det J_{\mathbf{x}|\mathbf{y} \mapsto \mathbf{z}}\right|\end{aligned}$$

Observation **y** selects from a family of networks that represent maps between p(**x**|**y**) and p(**z**)

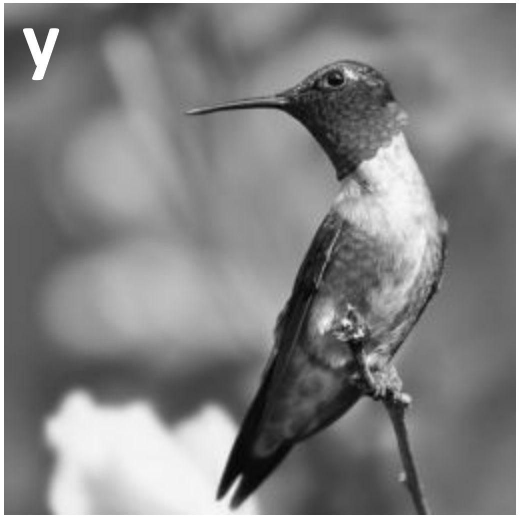
Conditional Coupling Block



Condition **c** as additional input to coupling network

Feature extractor can be shared between all blocks

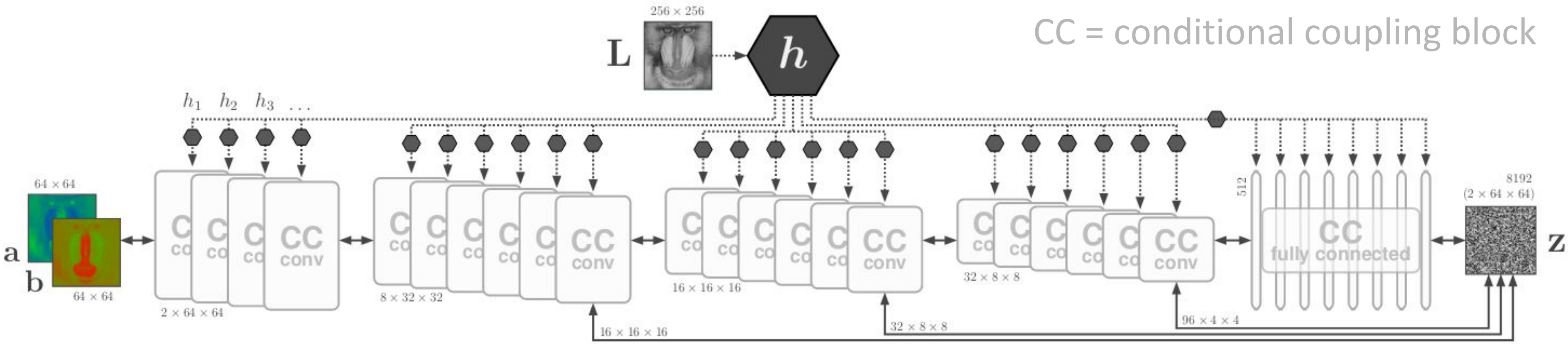
Higher dimensional example: Colorization



Inverse Problem:

Distribution over realistic color images **x** that look like **y** in grayscale

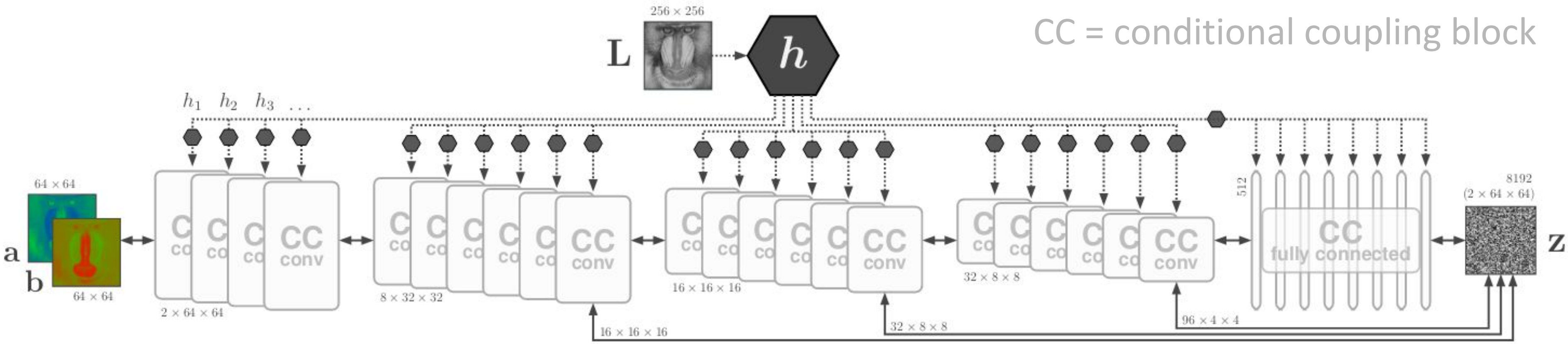
Higher dimensional example: Colorization



Four convolutional “stacks”, one fully connected

Random orthogonal matrices for mixing up channels

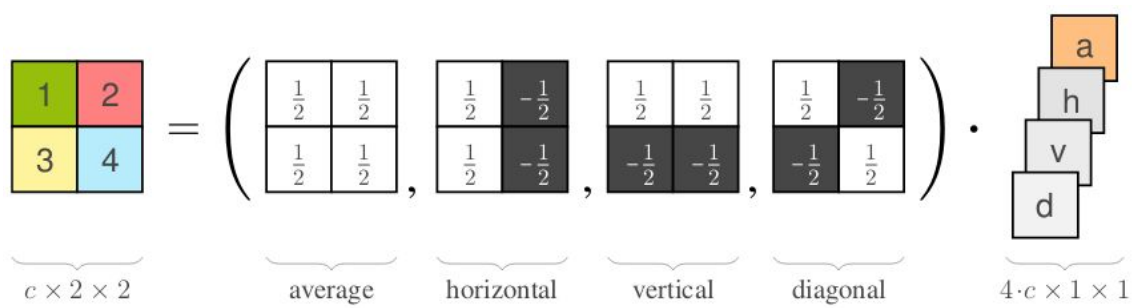
Higher dimensional example: Colorization



Four convolutional "stacks", one fully connected

Random orthogonal matrices for mixing up channels

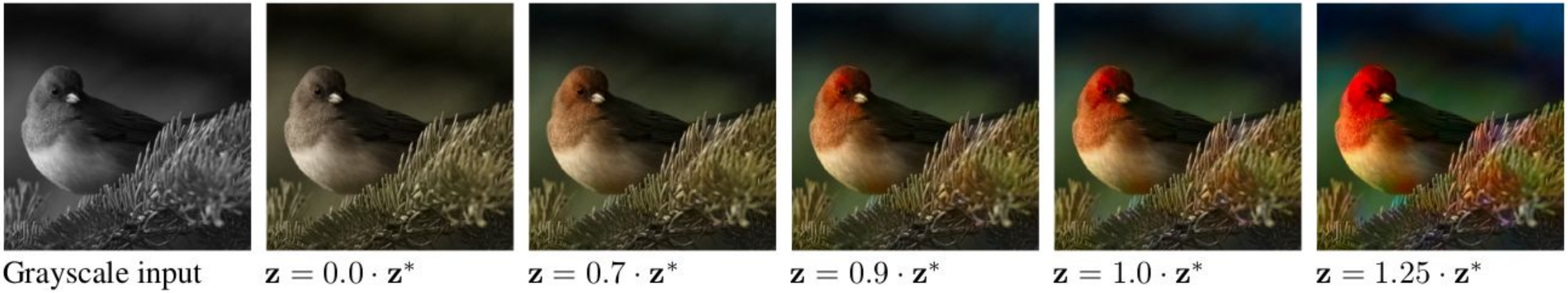
Multiscale via *Haar Wavelet* downsampling:



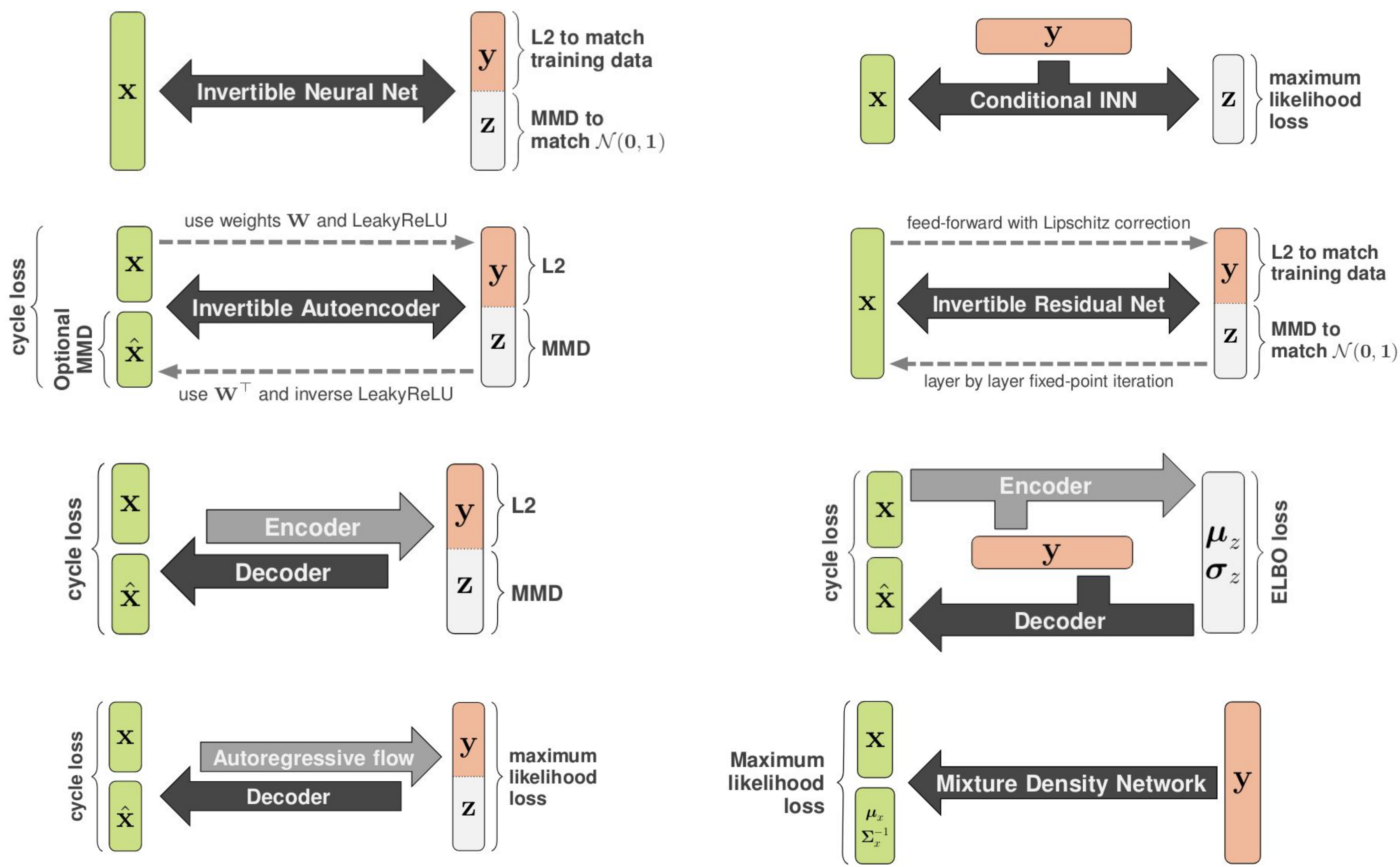
Meaningful latent space

Grayscale input | $\mathbf{z} = 0.0 \cdot \mathbf{z}^*$ | $\mathbf{z} = 0.7 \cdot \mathbf{z}^*$ | $\mathbf{z} = 0.9 \cdot \mathbf{z}^*$ | $\mathbf{z} = 1.0 \cdot \mathbf{z}^*$ | $\mathbf{z} = 1.25 \cdot \mathbf{z}^*$

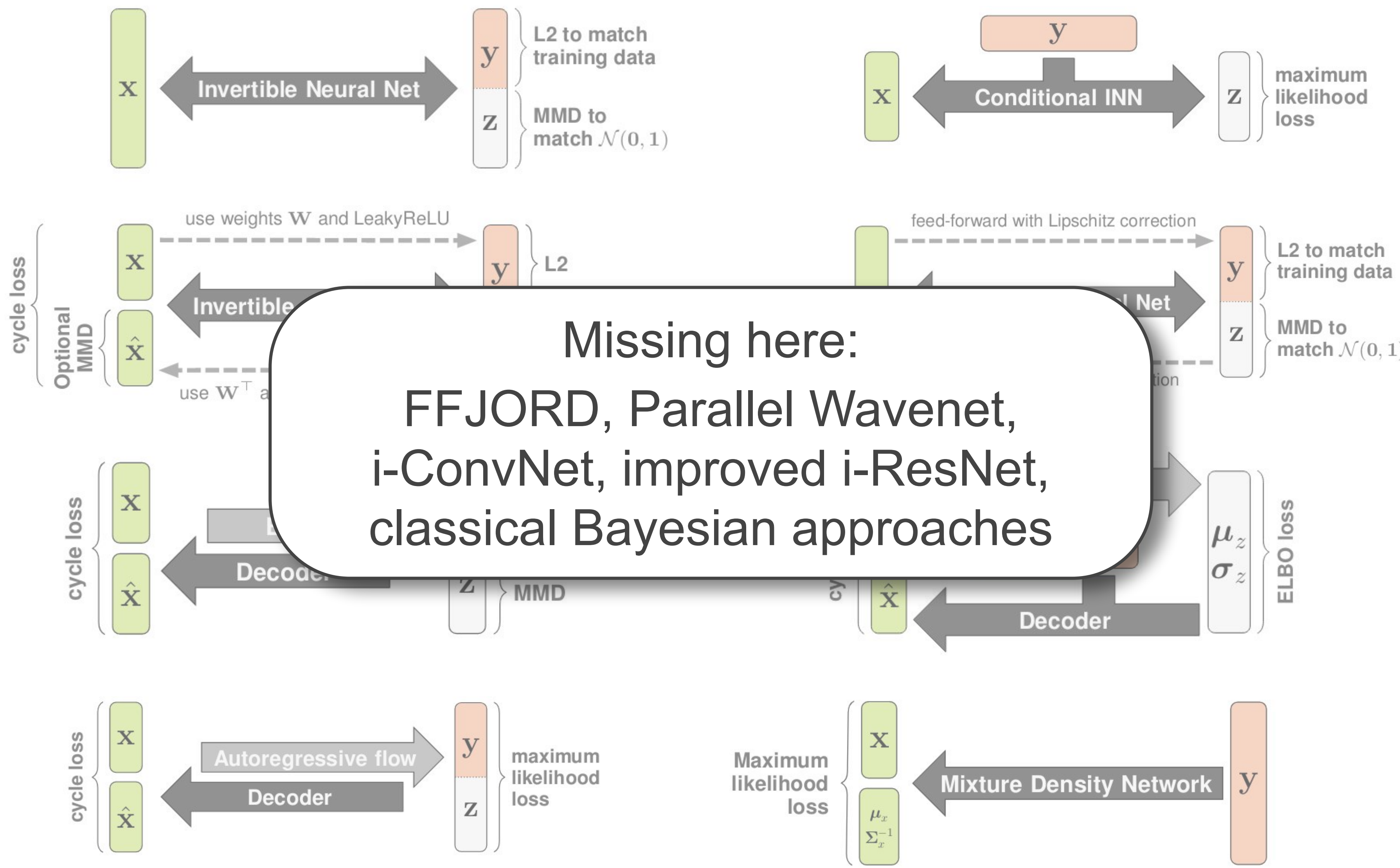
Meaningful latent space



Comparing architectures [2]



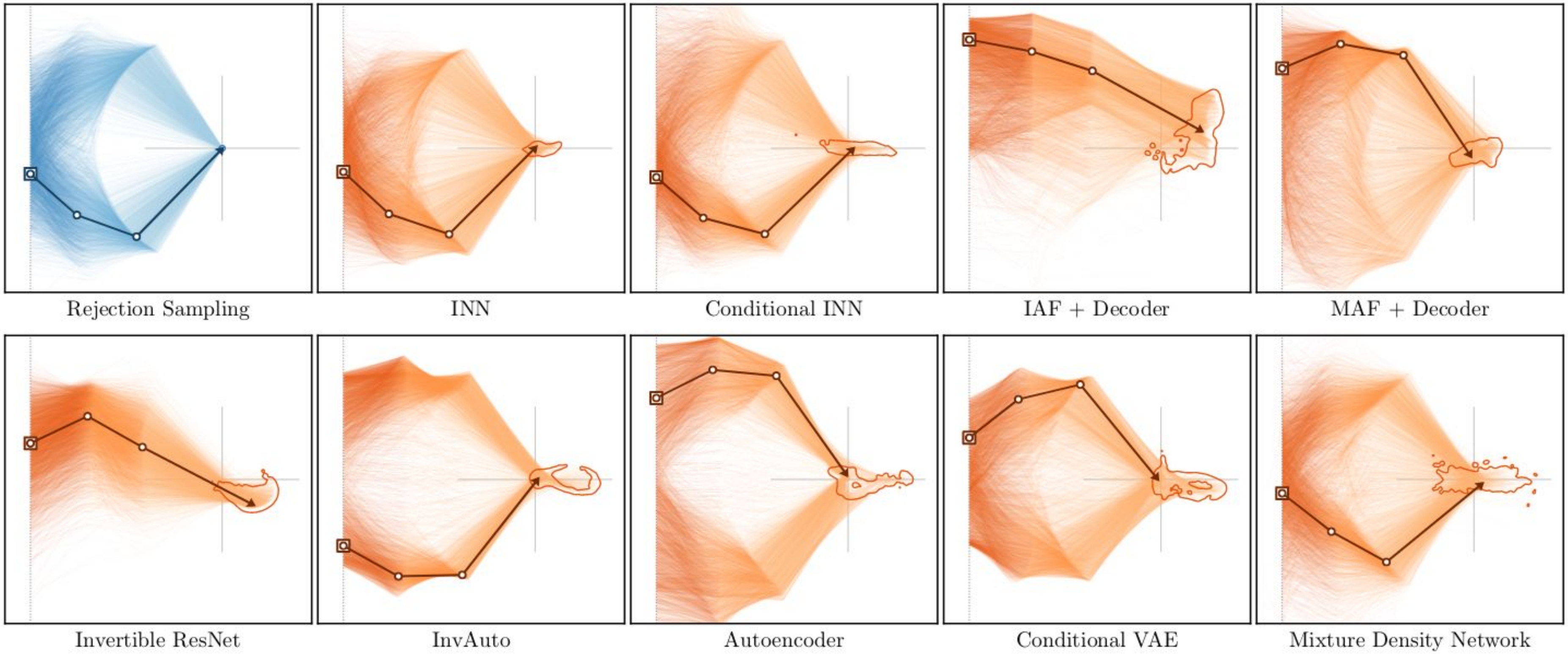
Comparing architectures [2]



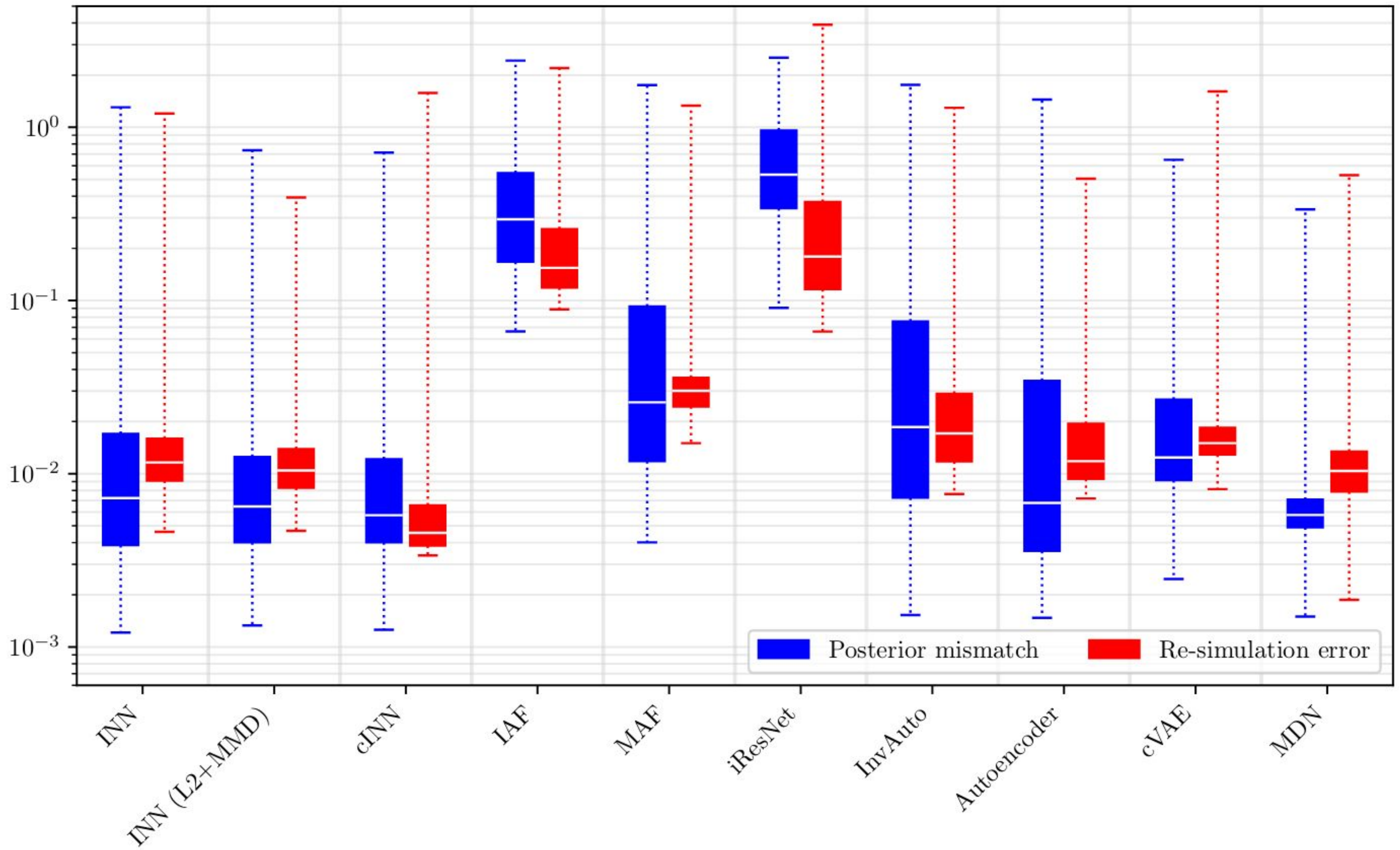
Missing here:

FFJORD, Parallel Wavenet, i-ConvNet, improved i-ResNet, classical Bayesian approaches

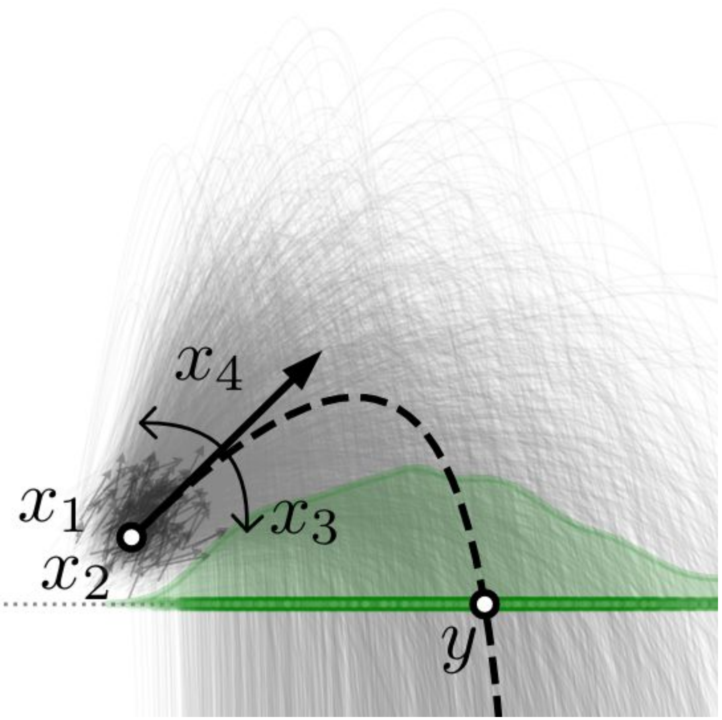
Comparing architectures: kinematics example

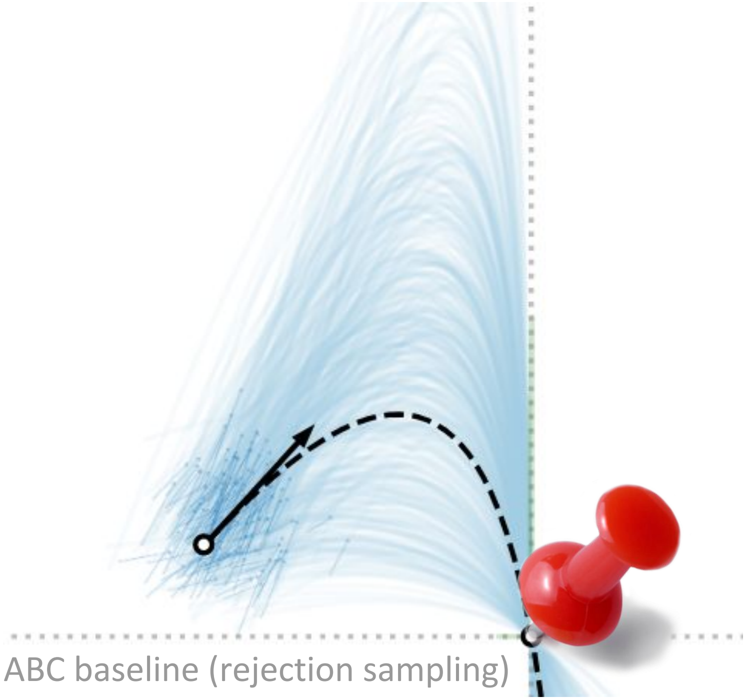


Comparing architectures: kinematics example



Comparing architectures: ballistics example

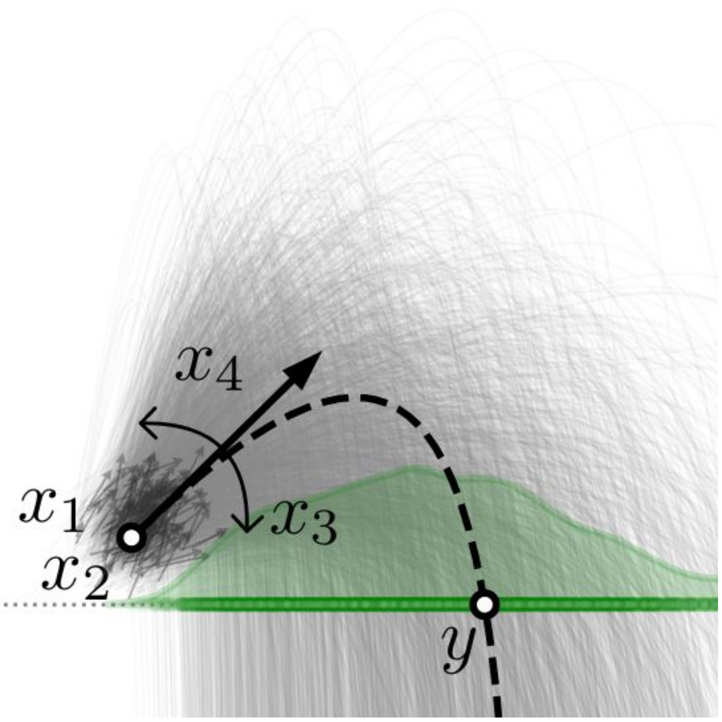


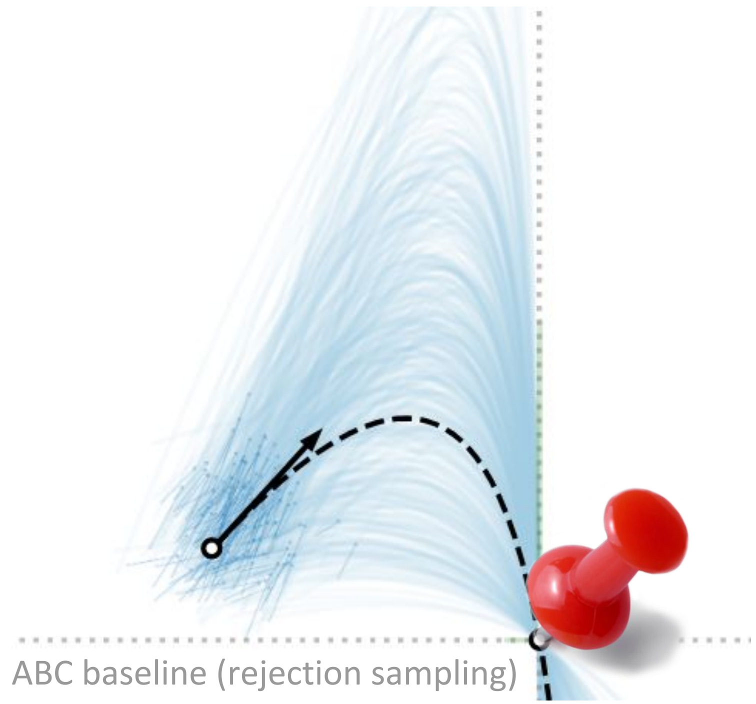


2D point mass thrown with gravity and drag

x is starting point, angle and force; y is impact location

Comparing architectures: ballistics example

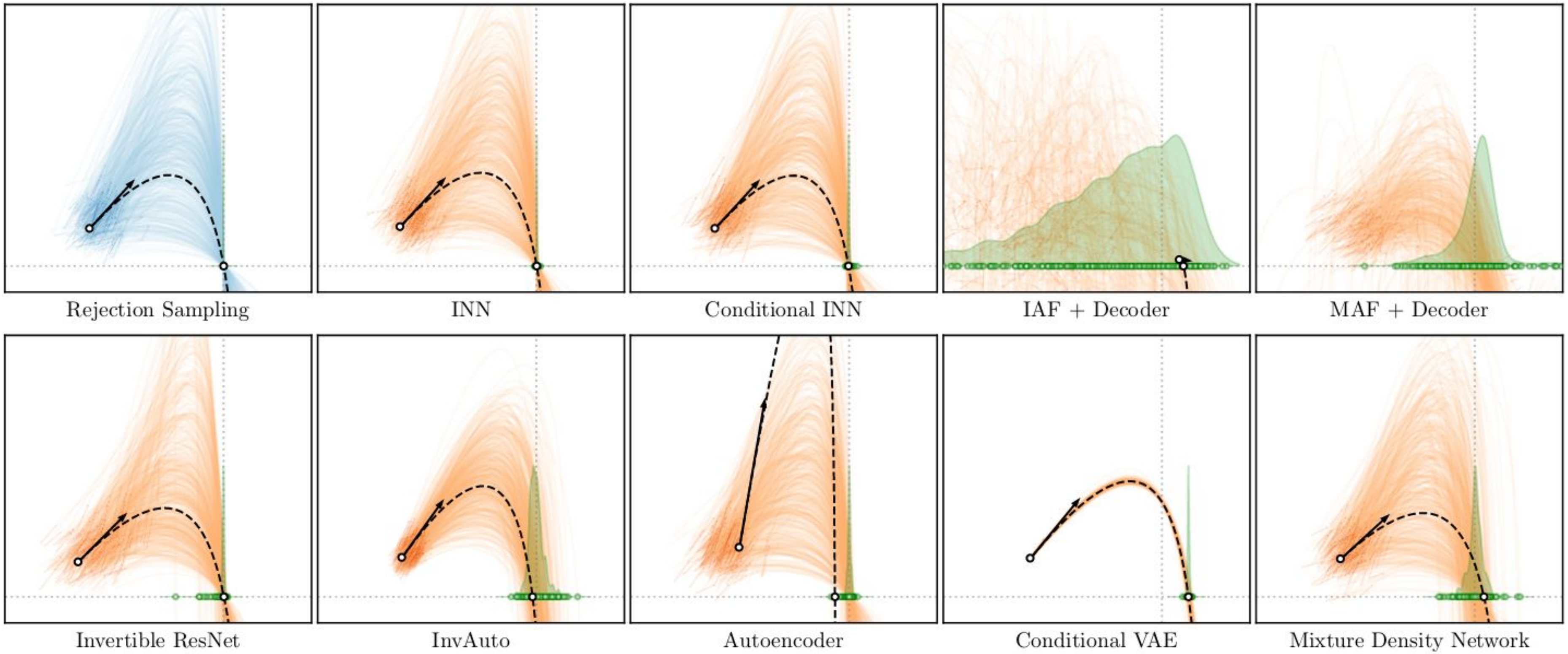




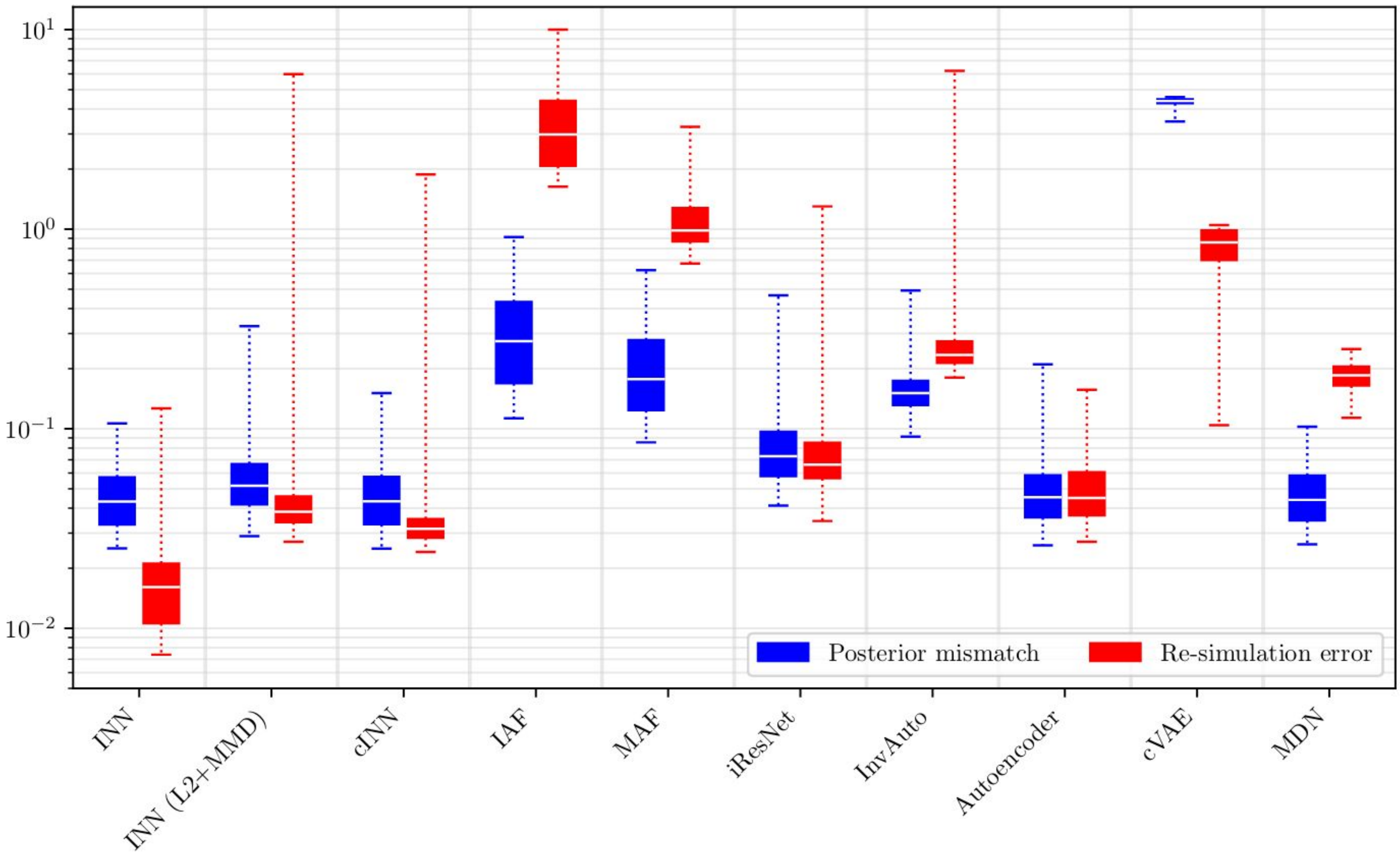
Inverse problem:

Distribution over all throws that hit a given location

Comparing architectures: ballistics example



Comparing architectures: ballistics example



Summary

INNs let us sample directly from the conditional posterior p(**x**|**y**) of an inverse problem

Two setups:
Direct (INN), with loss on forward process
Bayesian (cINN), where **y** is a conditional input

Several architectures; coupling blocks work well for us

Public code available under **github.com/VLL-HD/FrEIA**
(also contains many other invertible building blocks)

References

[1] Ardizzone, Lynton et al. **“Analyzing inverse problems with invertible neural networks”**. ICLR 2018.

[2] Kruse, Jakob et al. **“Benchmarking Invertible Architectures on Inverse Problems”**. ICML 2019 (INNF workshop).

[3] Ardizzone, Lynton et al. **“Guided Image Generation with Conditional Invertible Neural Networks”**. arXiv:1907.02392, under submission.

[4] Dinh, Laurent et al. **“Density estimation using Real NVP”**. ICLR 2017.

[5] Gretton, Arthur et al. **“A kernel two-sample test”**. Journal of Machine Learning Research 2012.

[6] Behrmann, J. et al. **“Invertible Residual Networks”**. ICML 2019.

[7] Chen, Tian Qi et al. **“Neural Ordinary Differential Equations”**. NIPS 2018.

[8] Teng, Yunfei et al. **“Invertible Autoencoder for domain adaptation”**. Computation 2018.

[9] Finzi, Mark et al. **“Invertible Convolutional Networks”**. ICML 2019 (INNF workshop).

[10] v.d. Oord, Aaron et al. **“ParallelWavenet: Fast High-Fidelity Speech Synthesis”**. arXiv:1711.10433.