Introduction to Normalizing Flows

Ullrich Köthe

Visual Learning Lab, Heidelberg University

joint work with **Lynton Ardizzone, Stefan Radev, Jakob Kruse, Peter Sorrenson, Tim Adler, Sebastian Wirkert, Victor Ksoll, Anja Butter, Armand Rousselot, Tilman Plehn, Ralf Klessen, Carsten Rother, Lena Maier-Hein**

Terascale School, March 2021

Generative Modelling

- Deep learning success story
  - Compute predictions y directly from complex data x
  - Point estimates: $\hat{y} \approx y^* = \operatorname{argmax} p(y \mid x)$, posteriors: $\hat{p}_\theta(y \mid x) \approx p(y \mid x)$
  - Relies on **discriminative / transductive machine learning**
    (does not first build a "model of the world" as traditional sciences do)
- Problem: discriminative models are hard to interpret, explain, validate

⇨ Generative modelling

- Turn the problem around: learn the data generation likelihood $p(x \mid y)$
- More difficult: requires ***insight*** beyond mere prediction capability
- Solve the original task via Bayes theorem

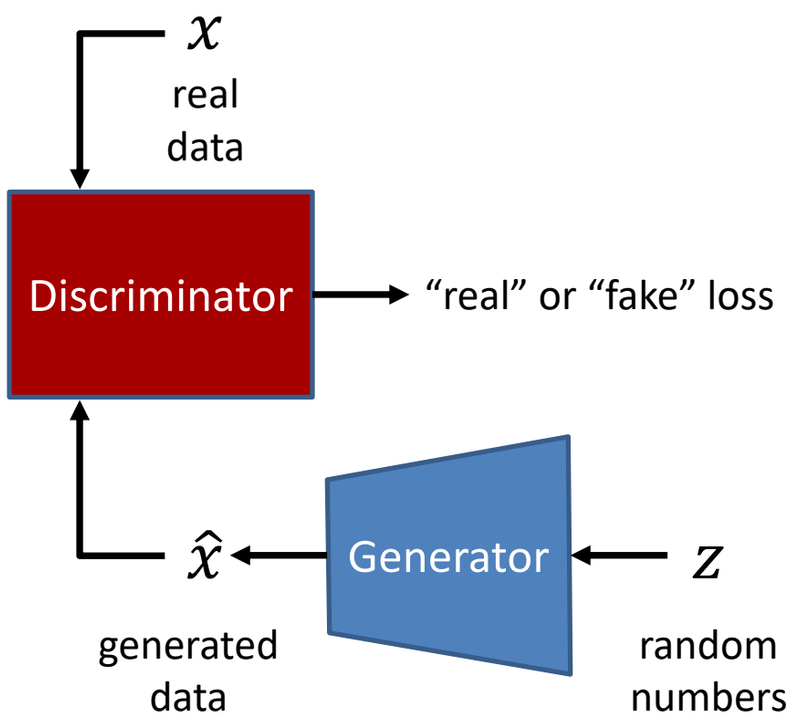
$$p(y \mid x) = \frac{p(x \mid y)\, p(y)}{p(x)}$$

Feynman: "What I cannot create, I do not understand."

Generative Modelling with Neural Networks

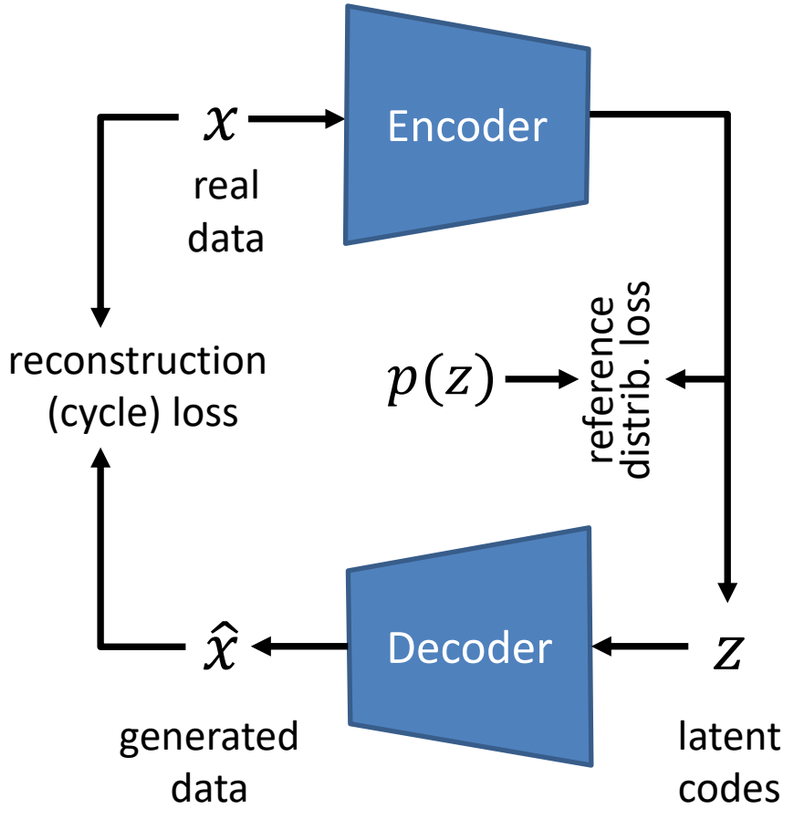
GANs

(Generative Adversarial Networks)



pure generation

(Variational) Autoencoders



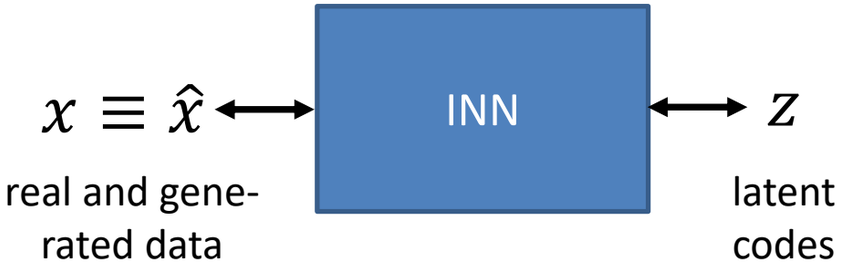
lossy encoding / decoding

Normalizing Flows

(Invertible Neural Networks, INNs)

maximum likelihood loss

$$p(x) = p\big(z = f(x)\big) \cdot |\det \nabla f|$$



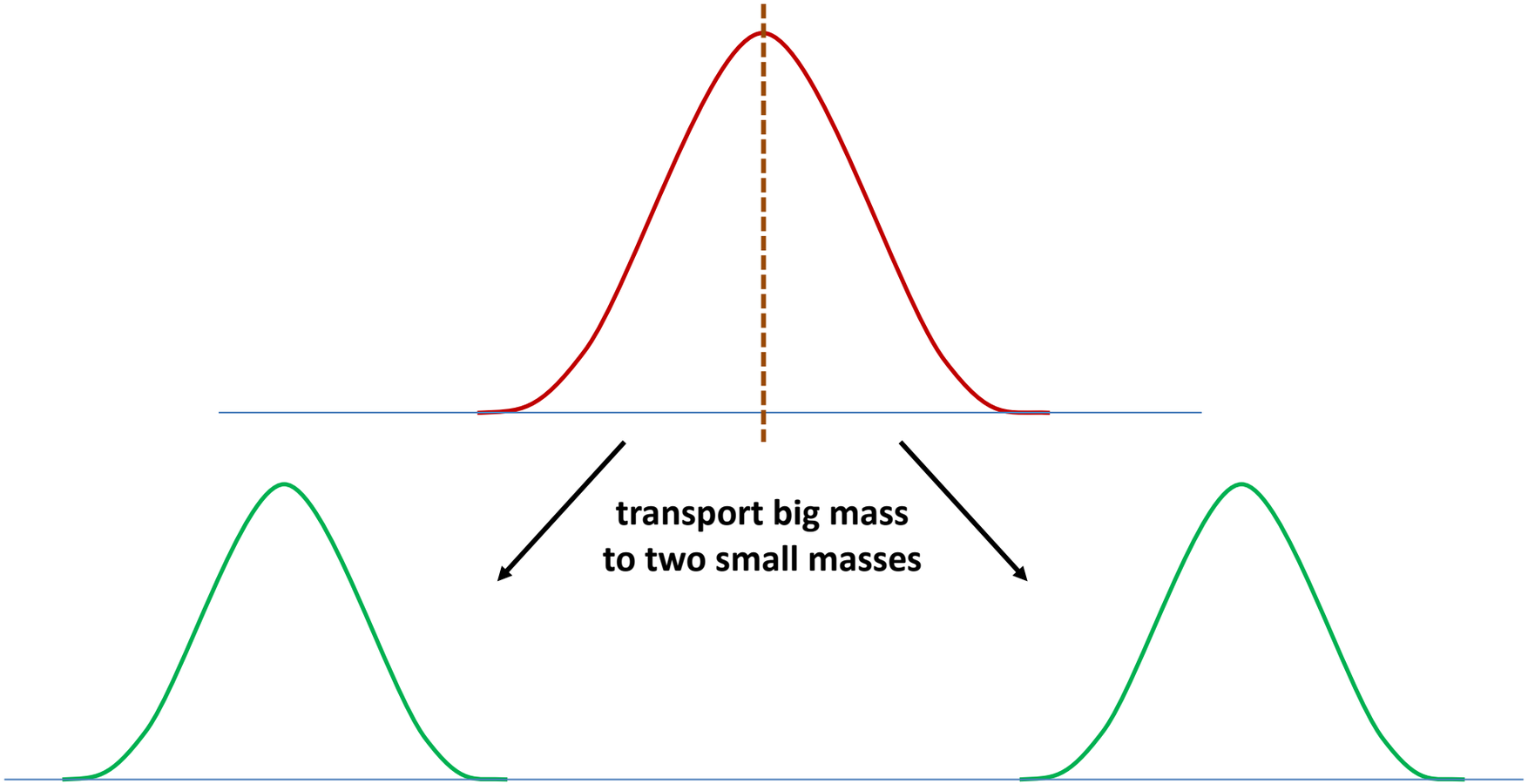
Encoder and Decoder
the are same network,
run forward / backward

lossless encoding / decoding

Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

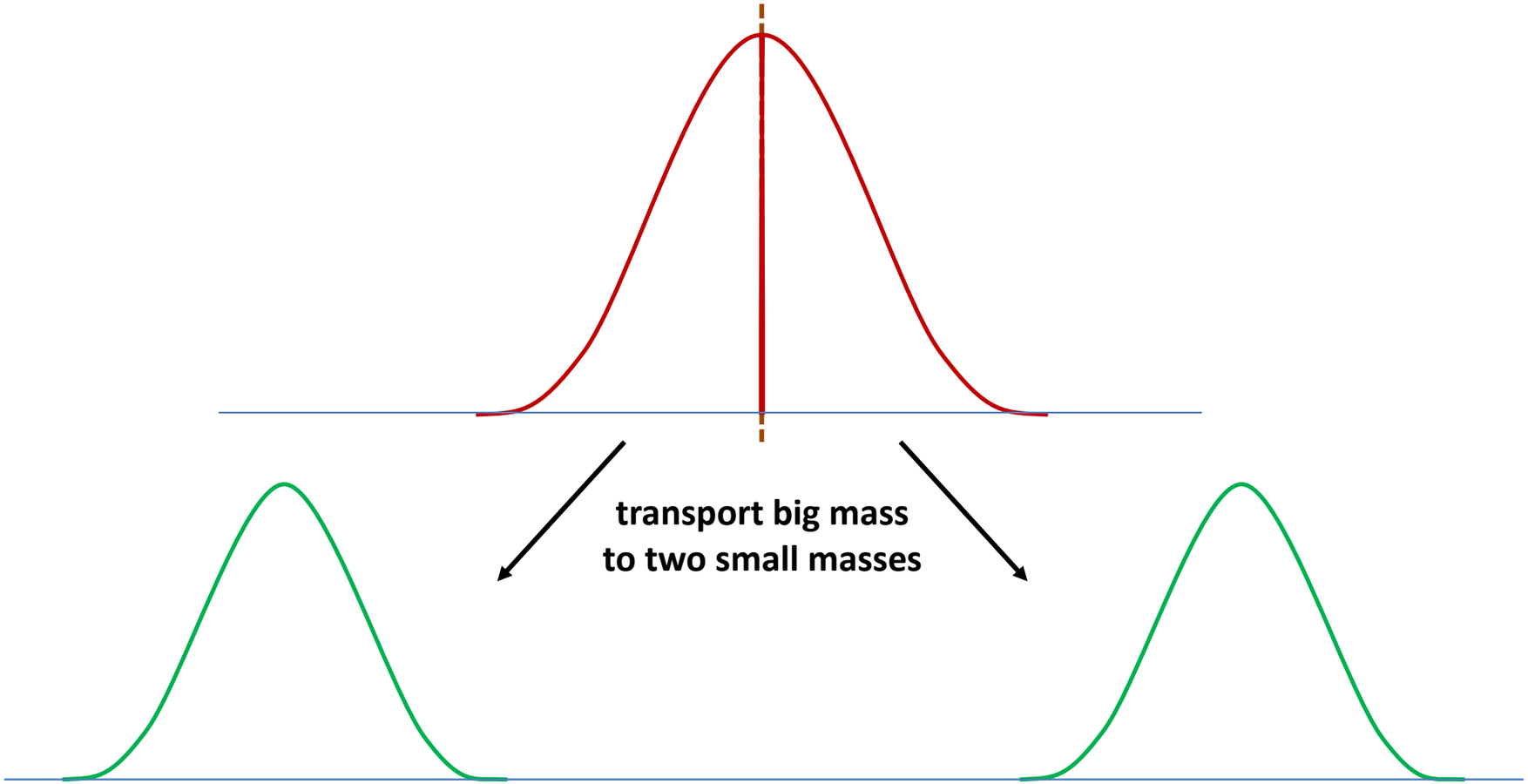
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

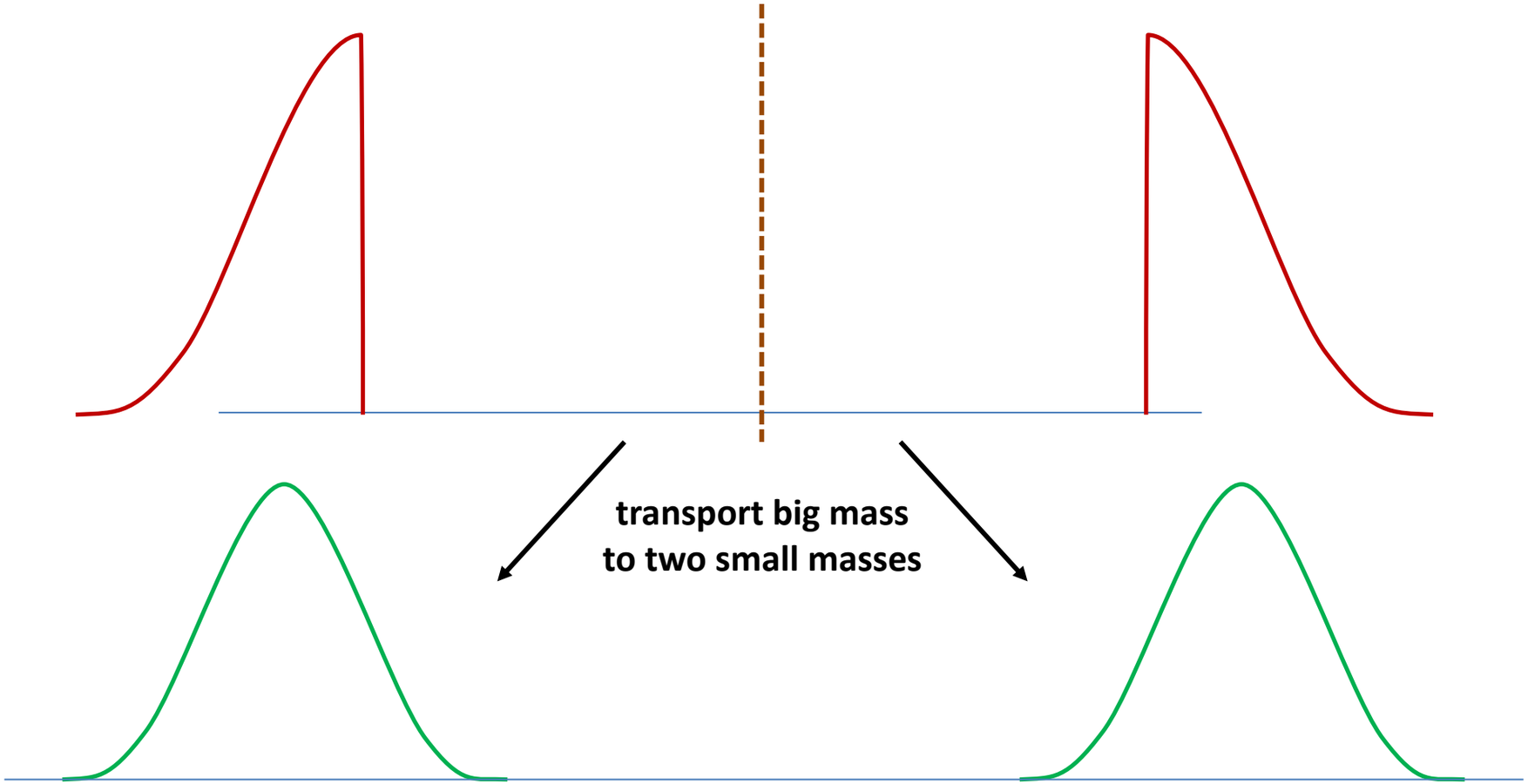
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

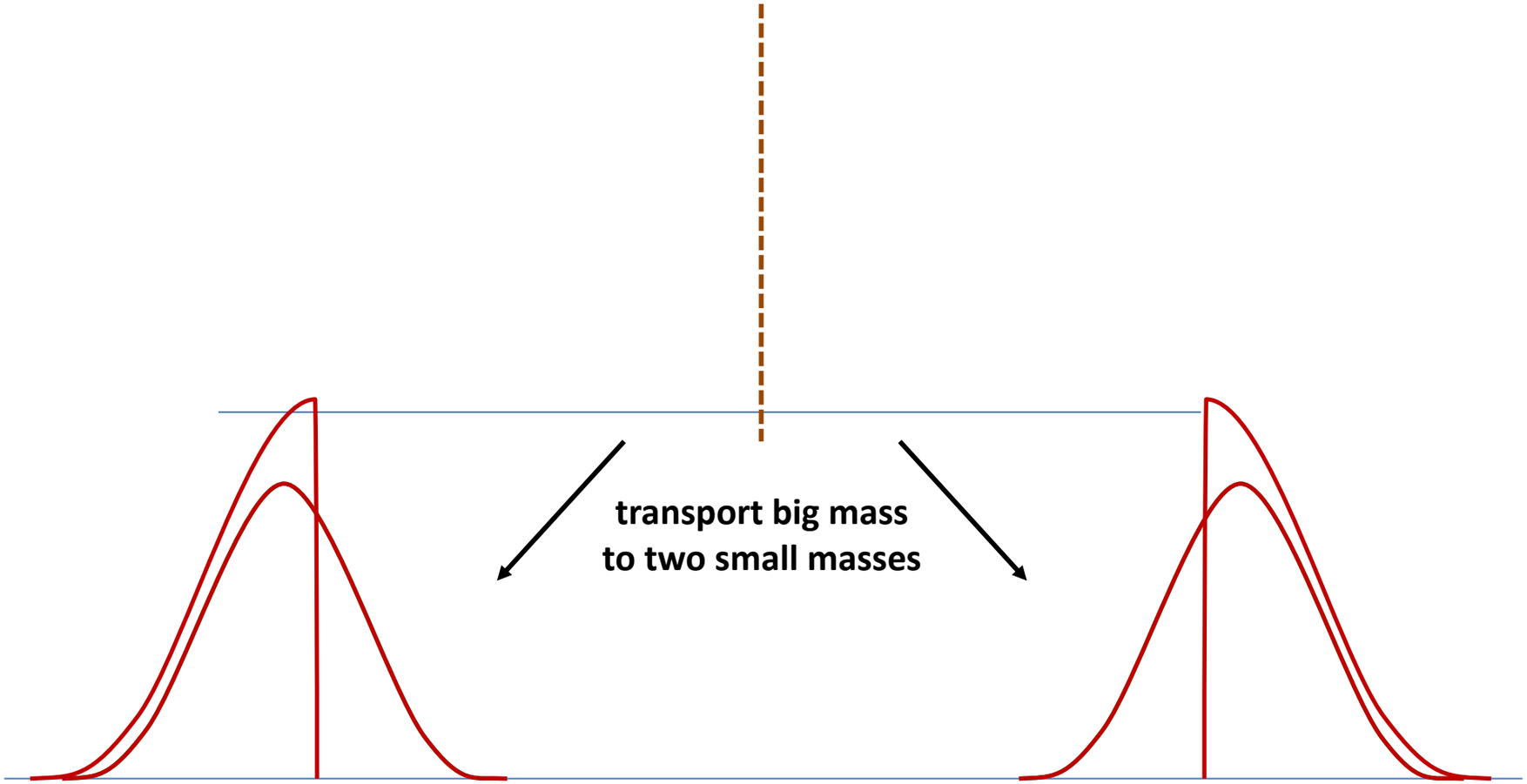
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



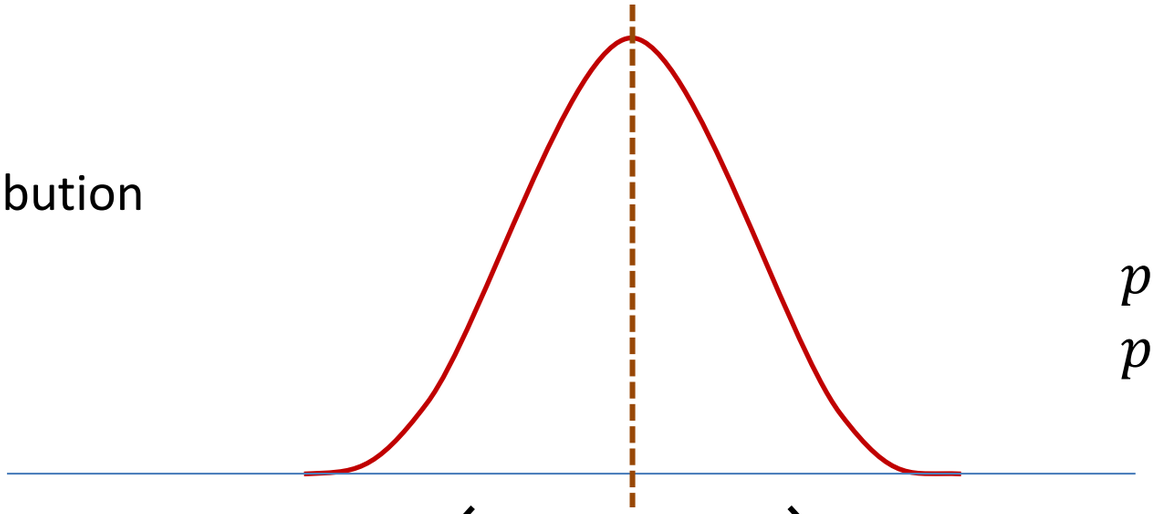
Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

- Mathematically: target distribution is a push-forward of reference distribution

reference distribution

$p(z)$



change of variables formula

$$p(x) = p\big(z = f(x)\big)\left|\det \nabla f\right|$$

$$p(x) = f_{\#}p(z)$$

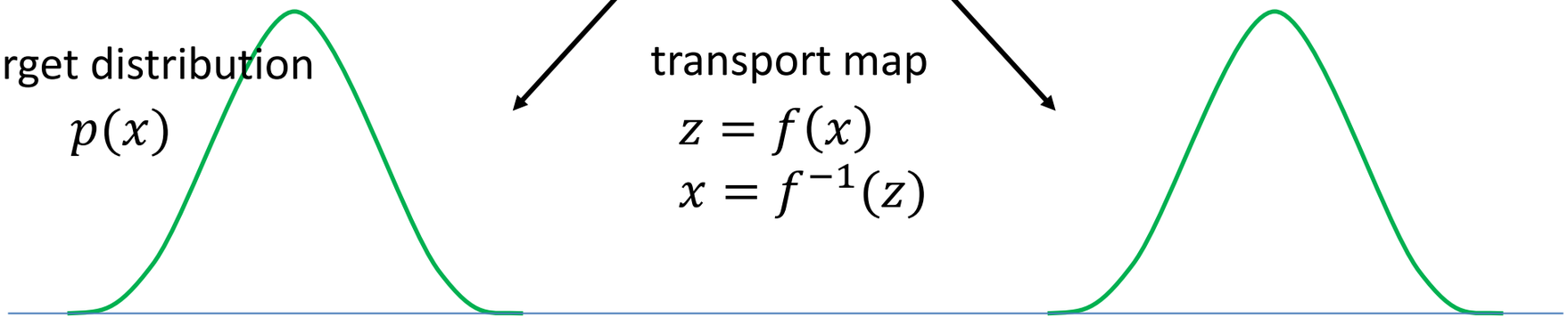
target distribution

$p(x)$

transport map

$z = f(x)$

$x = f^{-1}(z)$



Multiple Possibilities for Normalizing Flows

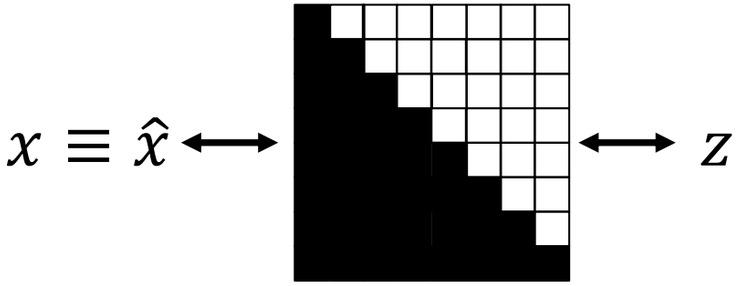
Autoregressive Models

Chain rule decomposition:

$$p(x_1, \dots, x_D) = \prod_i p_i(x_i \mid x_{<i})$$

triangular reparameterization:

$\forall i$: $x_i = f_i(z_i, x_{<i})$ monoton.



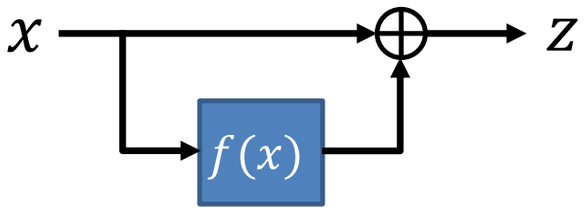
inverse direction inefficient

⇨ use two complementary nets

example: parallel WaveNet

iResNets
(invertible residual networks)

Residual block:



$$z = x + f(x)$$

is invertible when

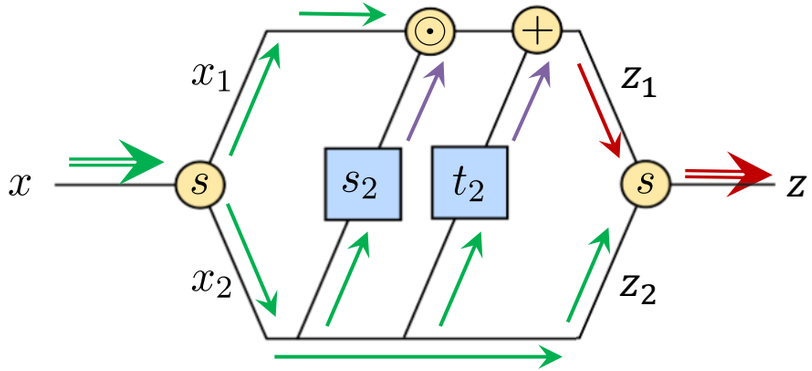
$$\|f(x)\|_{\text{Lipshitz}} < 1$$

inverse direction is reasonably efficient (fixpoint or Newton iterations)

example: Residual Flow Net

RealNVP

Affine coupling layer:



$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \cdot s_2(x_2) + t_2(x_2) \\ x_2 \end{bmatrix}$$

inverse is equally efficient:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (z_1 - t_2(z_2))/s(z_2) \\ z_2 \end{bmatrix}$$

example: GLOW

Coupling Layers and RealNVP

Invertible Neural Networks (INNs) with Coupling Layers

Powerful generative models: RealNVP („non-volume preserving“) [Dinh et al. 2017]

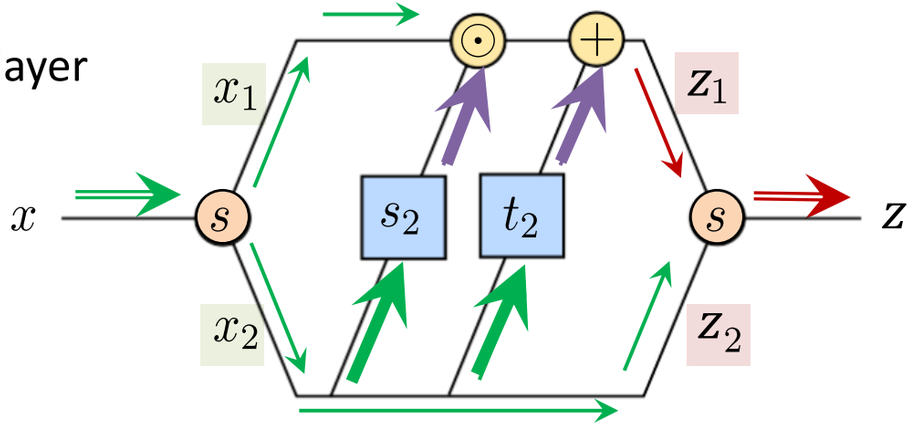
- Network is a sequence of *affine coupling layers*
- Each coupling layer splits its input $x \in \mathbb{R}^D$ into two halves $x_1, x_2 \in \mathbb{R}^{D/2}$
- Upper half is subjected to an affine transformation ⇨ outputs $z_1, z_2 \in \mathbb{R}^{D/2}$
- Affine coefficients are computed by standard fully connected or convolutional networks $s_2 \in \mathbb{R}_+^{D/2}$ and $t_2 \in \mathbb{R}^{D/2}$ from the lower half's data

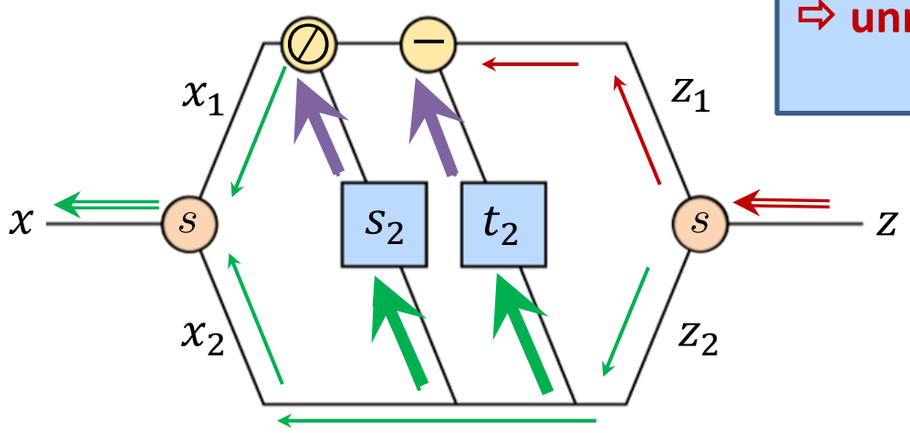
Forward computation: $z_1 = x_1 \odot s_2(x_2) + t_2(x_2), \quad z_2 = x_2$

Inverse computation: $x_1 = (z_1 - t_2(z_2)) \oslash s_2(z_2), \quad x_2 = z_2$

nested functions s_2 and t_2 are always executed in the same direction ⇨ unrestricted neural networks

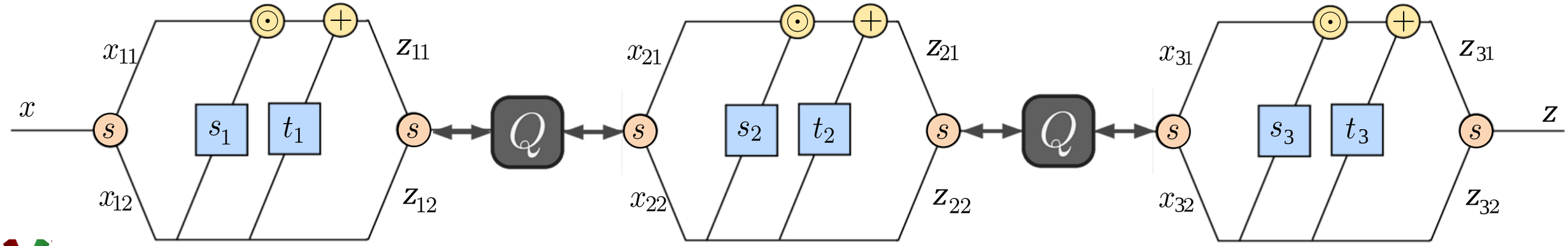
Coupling layer



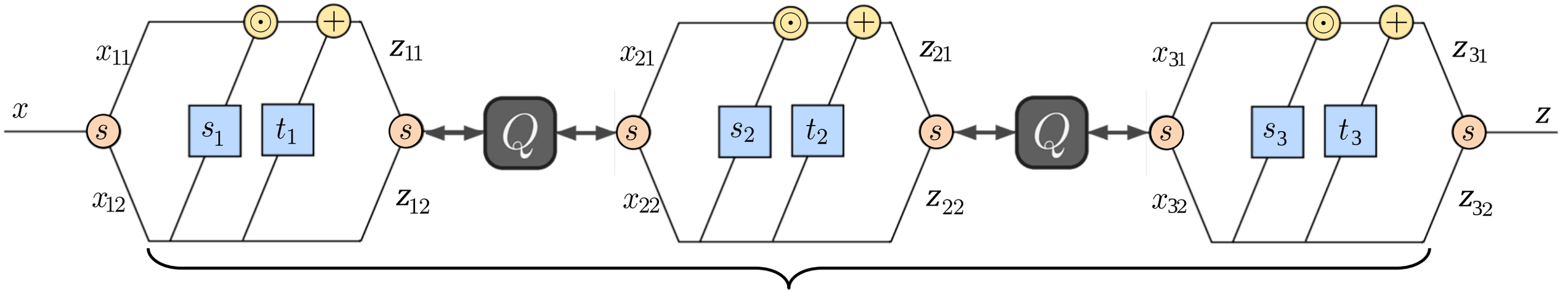


Deep INNs

- Concatenate many coupling layers
- Alternate with orthogonal layers Q
  - ⇨ Active (upper lane) and passive (lower lane) dimensions change in each layer
  - – Random permutations or projections are good enough, learning Q is not necessary
- Surprisingly powerful despite its simplicity
- Similar to autoencoder: forward mode = encoder, backward mode = decoder
  - – Encoder and decoder are merged into a single network
  - – Lossless encoding due to invertibility (no bottleneck)



Training Deep INNs with Maximum Likelihood Loss



- Training objective: $\hat{\theta} = \arg\max_\theta \prod_i p_\theta\left(x^{(i)}\right)$

$$= \arg\min_\theta \sum_i -\log p\left(z = f_\theta\left(x^{(i)}\right)\right) - \log\det\nabla f_\theta\left(x^{(i)}\right)$$

- Especially simple when $p(z)$ is standard normal and $s_l(x_{l2}) = \exp\tilde{s}_l\left(x_{l2}\right)$:

$$\hat{\theta} = \arg\min_\theta \sum_i \left[f_\theta\left(x^{(i)}\right)^2 - 2\sum_l \left\|\tilde{s}_l\left(x_{l2}^{(i)}\right)\right\|_1\right]$$

Invertible ResNets

Recap: What is a ResNet?

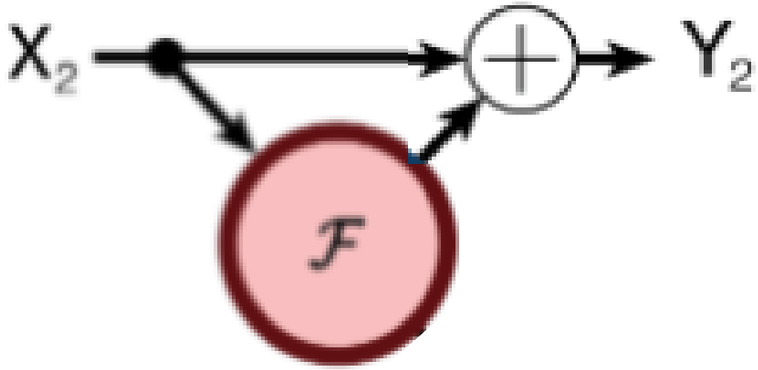
- Instead of modeling the transition from layer l to $l+1$

$$z_{l+1} = \mathcal{F}_l(z_l)$$

model the difference (residual) between consecutive layers

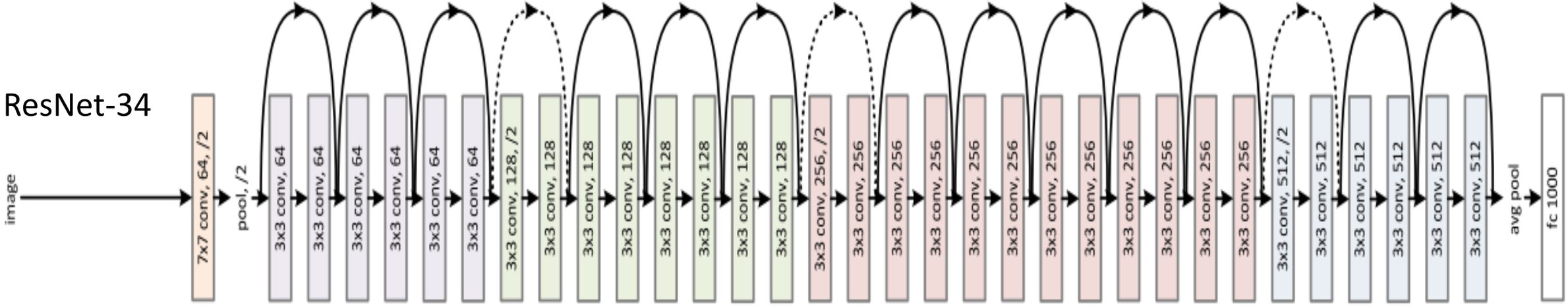
$$z_{l+1} - z_l = \mathcal{F}_l(z_l) \quad \Leftrightarrow \quad z_{l+1} = z_l + \mathcal{F}_l(z_l)$$

  - Each layer ("residual block") consists of a skip connection and a parallel feed-forward transformation
  - Advantage: no vanishing gradients even for very deep networks



residual block

ResNet-34

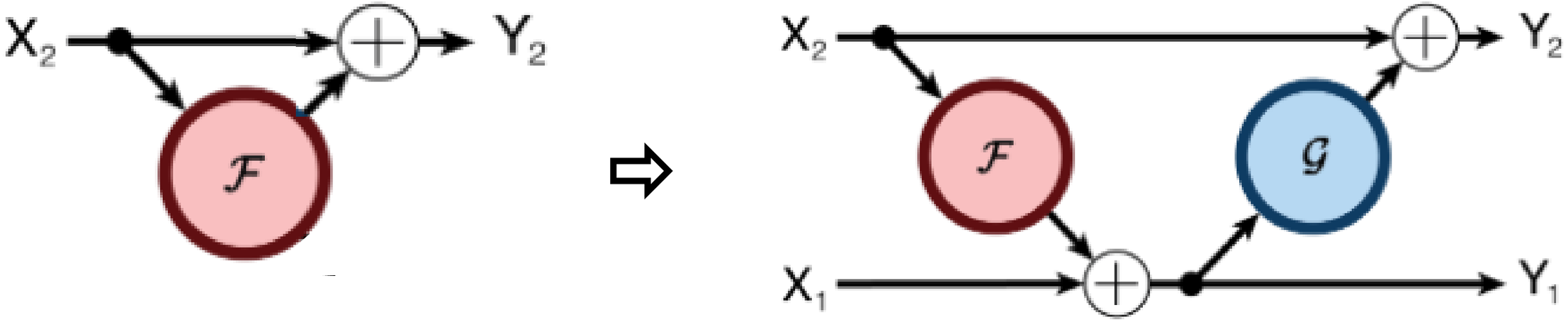


He et al. **"Deep residual learning for image recognition"**, CVPR 2016.

RevNets: Memory-efficient backpropagation

- Simple application of coupling layers: replace residual blocks with coupling blocks
  - Do not store activations during the forward pass of training
  - Recompute them on the fly during backpropagation, using the invertible architecture

Residual block becomes: Additive coupling block



RevNets: Memory-efficient backpropagation

- Simple application of coupling layers: replace residual blocks with coupling blocks
  - Do not store activations during the forward pass of training
  - Recompute them on the fly during backpropagation, using the invertible architecture

Algorithm 1 Reversible Residual Block Backprop

function BLOCKREVERSE($(y_1, y_2), (\overline{y}_1, \overline{y}_2)$)

$z_1 \leftarrow y_1$

$x_2 \leftarrow y_2 - \mathcal{G}(z_1)$

$x_1 \leftarrow z_1 - \mathcal{F}(x_2)$

inverse pass

$\overline{z}_1 \leftarrow \overline{y}_1 + \left(\frac{\partial \mathcal{G}}{\partial z_1}\right)^\top \overline{y}_2$

$\overline{x}_2 \leftarrow \overline{y}_2 + \left(\frac{\partial \mathcal{F}}{\partial x_2}\right)^\top \overline{z}_1$

$\overline{x}_1 \leftarrow \overline{z}_1$

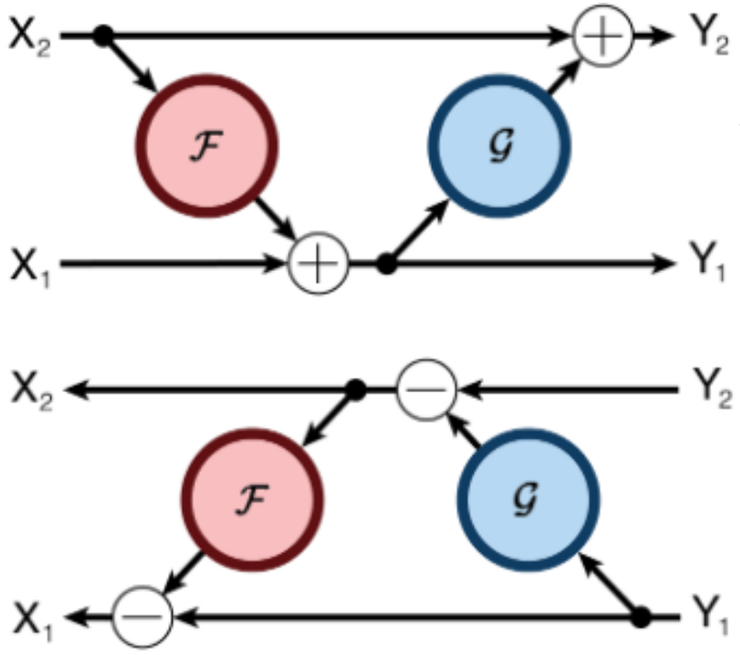
$\overline{w}_\mathcal{F} \leftarrow \left(\frac{\partial \mathcal{F}}{\partial w_\mathcal{F}}\right)^\top \overline{z}_1$

$\overline{w}_\mathcal{G} \leftarrow \left(\frac{\partial \mathcal{G}}{\partial w_\mathcal{G}}\right)^\top \overline{y}_2$

"isolated" autodiffs of G and F

return (x_1, x_2) and $(\overline{x}_1, \overline{x}_2)$ and $(\overline{w}_\mathcal{F}, \overline{w}_\mathcal{G})$

end function



Gomez & Ren et al. **"The Reversible Residual Network: Backpropagation Without Storing Activations"**, NIPS 2017.

RevNets: Memory-efficient backpropagation

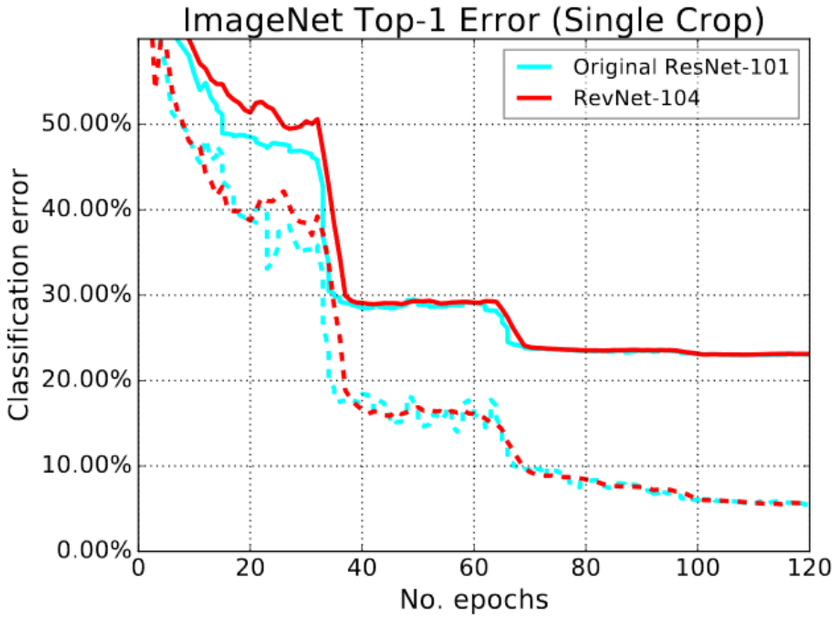
- Performance example: ResNet-101 vs. RevNet-104 on ImageNet

Dataset	Version	Params (M)	Units	Parameter Cost	Activation Cost
ImageNet	ResNet-101	44.5	3-4-23-3	~ 178MB	~ 5250MB
ImageNet	RevNet-104	45.2	2-2-11-2	~ 180MB	~ 1440MB

- Very similar behavior:

Top-1 classification error

ResNet-101	RevNet-104
23.01%	23.10%



– Trade-off: greatly reduces memory consumption for 2-4 times the compute

Gomez & Ren et al. **"The Reversible Residual Network: Backpropagation Without Storing Activations"**, NIPS 2017.

Application: i-RIM 3D

- Allows training of very big nets: 3-dimensional convolutions, many layers
  - fastMRI Challenge: MRI reconstruction from 8x less raw data

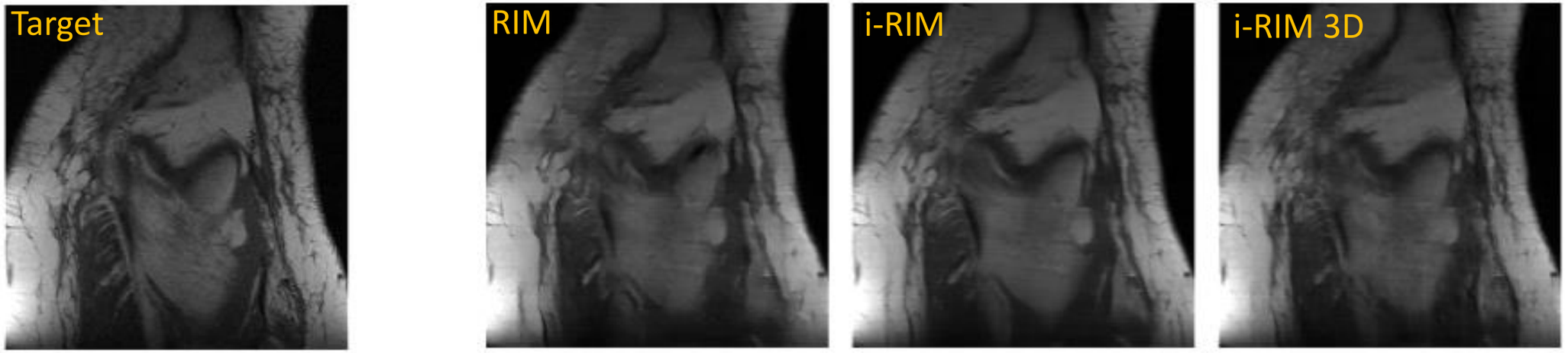


Table 1: Comparison of memory consumption during training and testing.

	RIM	i-RIM 2D	i-RIM 3D
Size Machine State $(\boldsymbol{\eta}, \mathbf{s})$ (CDHW)	$130 \times 1 \times 480 \times 320$	$64 \times 1 \times 480 \times 320$	$64 \times 32 \times 480 \times 320$
Memory Machine State $(\boldsymbol{\eta}, \mathbf{s})$ (in GB)	0.079	0.039	1.258
Number of steps T	1/4/8	1/4/8	1/4/8
Network Depth (#layers)	5/20/40	50/200/400	50/200/400
Memory during Testing (in GB)	0.60 / 0.65 / 0.65	0.20 / 0.24 / 0.31	5.87/6.03 / 6.25
Memory during Training (in GB)	2.65 / 6.01 / 10.49	2.47 / 2.49 / 2.51	11.51 / 11.76 / 11.89

Making ResNets Invertible: i-ResNets and Residual Flows

- Can one create an invertible network while keeping the original ResNet architecture?

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

  - How to ensure a bijective mapping?
  - How to compute the inverse efficiently?
  - How to perform maximum likelihood training?

- The mapping is guaranteed to be bijective if $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t} > 0$
  - Sufficient condition: Lipschitz bound on g_{θ_t}: $\left\| g_{\theta_t}\left(x_t^{(1)}\right) - g_{\theta_t}\left(x_t^{(2)}\right) \right\| \leq \lambda \left\| x_t^{(1)} - x_t^{(2)} \right\|$ with $\lambda < 1$

⇨ Expressive power of each block is limited, need more blocks

⇨ Blocks can be inverted using fixed point iterations or Newton's method:

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

Behrmann et al. **"Invertible Residual Networks"**, ICML 2019.

Making ResNets Invertible: i-ResNets and Residual Flows

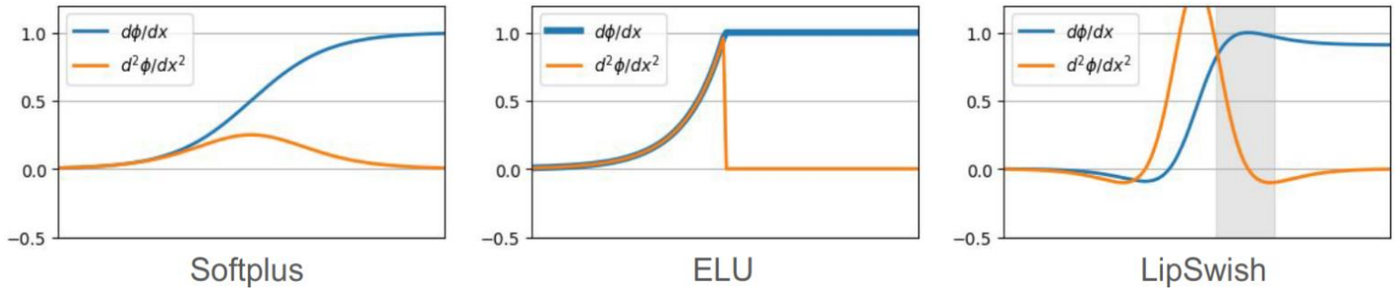
- How to achieve the Lipschitz bound in a layer $x_{i+1} = x_i + \phi(W x_i)$?
  - Step 1: normalize weight matrices with $c < 1$ and largest singular value $\tilde{\sigma}_i \leq \|W_i\|_2$ estimated by (one iteration of) power method

$$\tilde{W}_i = \begin{cases} c\, W_i / \tilde{\sigma}_i, & \text{if } c/\tilde{\sigma}_i < 1 \\ W_i, & \text{else} \end{cases}$$

Randomly initialise $\vec{x}_0$
for $i = 1$ **to** n **do**
  $\vec{x}_i \leftarrow W^T W \vec{x}_{i-1}$
end for
$\sigma_{max} \leftarrow \frac{\|W\vec{x}_n\|_2}{\|\vec{x}_n\|_2}$

  - Step 2: use suitable activation functions: $\forall r: \ |\phi'(r)| \leq 1$
    - fulfilled by many $\phi(r)$, **but** training involves the gradient of the log-determinant of the Jacobian (the first derivative), i.e. the *second* derivative $\phi''(r)$
  - Many common $\phi(r)$ have $\phi'(r) \approx 1 \Rightarrow \phi''(r) \approx 0$, i.e. suffer from vanishing gradients

⇨ Choose $\phi(r) = \text{LipSwish}(r) = 0.909\, r/(1 + \exp(-\beta\, r))$



Gouk et al. **"Regularisation of Neural Networks by Enforcing Lipschitz Continuity"**, arXiv 2018.
Chen et al. **"Residual Flows for Invertible Generative Modeling"**, NeurIPS 2019.

Making ResNets Invertible: i-ResNets and Residual Flows

- Improvements of Residual Flow over i-ResNet apparent visually and in the numbers

Model	MNIST	CIFAR-10	ImageNet 32×32	ImageNet 64×64
Real NVP (Dinh et al., 2017)	1.06	3.49	4.28	3.98
Glow (Kingma and Dhariwal, 2018)	1.05	3.35	4.09	3.81
FFJORD (Grathwohl et al., 2019)	0.99	3.40	—	—
Flow++ (Ho et al., 2019)	—	**3.29** (3.09)	— (3.86)	— (3.69)
i-ResNet (Behrmann et al., 2019)	1.05	3.45	—	—
Residual Flow	**0.97**	**3.29**	**4.02**	**3.78**

Chen et al. **"Residual Flows for Invertible Generative Modeling",** NeurIPS 2019.

Application: Solving Inverse Problems

Sources of Uncertainty in Machine Learning

Goal: given an unknown true data generating mechanism

$$y \sim p^*(y \mid x)$$

find optimal approximation from a given model family $\mathcal{F}$

$$q^*(y \mid x) = \arg\min_{q \in \mathcal{F}} \mathcal{L}(q, p^*)$$

by minimizing a loss $\mathcal{L}(q, p^*)$ measuring the approximation error

Error sources for the deterministic case $y = g^*(x)$

- Model misspecification error: $\mathcal{F}$ does not contain g^*

$$f^*(x) = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f, g^*) \neq g^*(x)$$

- Epistemic error: g^* is only known via a finite random training set $\mathcal{TS}$

$$\tilde{f}(x) = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f, \mathcal{TS}) \neq f^*(x)$$

- Optimization error: numeric and algorithmic approximations

$$\hat{f}(x) = \text{algorithmic} \min_{f \in \mathcal{F}} \mathcal{L}(f, \mathcal{TS}) \neq \tilde{f}(x)$$

} now of lesser concern thanks to neural nets

Sources of Uncertainty in Machine Learning

Additional error sources for the probabilistic case $y \sim p^*(y \mid x)$

- Aleatoric error: observations x and mechanism p^* are noisy
  - classic ML: reduce to the deterministic case by defining

$$g^*(x) = \mathbb{E}_{y \sim p^*(y \mid x)}[y \mid x] \qquad \text{or} \qquad g^*(x) = \arg\max_y p^*(y \mid x)$$

- Ambiguity error: x does not contain enough information to fully recover y
  - classic ML: reduce to the deterministic case by regularization term $\mathcal{R}(f)$

$$\tilde{f}(x) = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f, \mathcal{TS}) + \lambda \cdot \mathcal{R}(f)$$

- Bayesian statistics allow recovery of full posterior $q^*(y \mid x) \approx p^*(y \mid x)$, but
  - classic parametric models often suffer from misspecification error
  - non-parametric models are often expensive

Invertible neural networks offer great new possibilities!

Simulation-Based Inference for Inverse Problems

- **Inverse problem goal:** given observations $\hat{y}$, determine underlying hidden parameters $\hat{x}$
- In many inverse problems, the forward process is well understood…
  - Differential equations
  - Markov chains
  - Monte-Carlo simulations
  - …

$$\boldsymbol{y} = g(\boldsymbol{x}; \boldsymbol{\xi})$$

with $\boldsymbol{x}$ the hidden parameters and $\boldsymbol{\xi}$ a noise vector (i.e. the random numbers used in the simulation)

… but inversion is still difficult:

- likelihood $p(\boldsymbol{y} \mid \boldsymbol{x})$ of observed data $\boldsymbol{y}$ in Bayes formula

$$p(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x})\, p(\boldsymbol{x})}{p(\boldsymbol{y})}$$

is only implicitly defined by the simulation $\boldsymbol{y} = g(\boldsymbol{x}; \boldsymbol{\xi})$

⇨ **likelihood cannot be evaluated, posterior is analytically intractable**

“likelihood-free inference”

Simulation-Based Inference

- Classical simplification: reduce posterior to point estimate
  - **Regularization:** disambiguate inverse when forward process is surjective (not information preserving)
  - Maximum a-posteriori (**MAP**) inference: find only mode of posterior
  - ⇨ No idea of solution diversity and uncertainty
- Standard solution: approximate Bayesian computation (ABC)
  - Define a distance $d(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{sim}})$ between observed and simulated data
    - for m=1,…, M:

      Sample hidden parameters $\boldsymbol{x}^{(m)} \sim p(\boldsymbol{x})$ from prior

      Run the simulation $\boldsymbol{y}_{\text{sim}}^{(m)} = g(\boldsymbol{x}^{(m)}; \boldsymbol{\xi})$

      Keep $\boldsymbol{x}^{(m)}$ if $d\left(\boldsymbol{y}_{\text{obs}}, \boldsymbol{y}_{\text{sim}}^{(\boldsymbol{m})}\right) < \epsilon$, reject otherwise
    - Return the set of "surviving" $\boldsymbol{x}^{(m)}$ as an approximate sample from $p(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$
  - ⇨ Very slow: high rejection rate, because small ϵ are needed for accurate results
- Case-based inference
  - Efficient sampling methods (MCMC, SMC, …) with good proposal distribution have low rejection rates
  - ⇨ Learn a good proposal distribution for each observed data set $\boldsymbol{y}_{\text{obs}}$, i.e. on a per case basis
  - ⇨ Still expensive if many different $\boldsymbol{y}_{\text{obs}}$ must be evaluated

Linear Toy Example

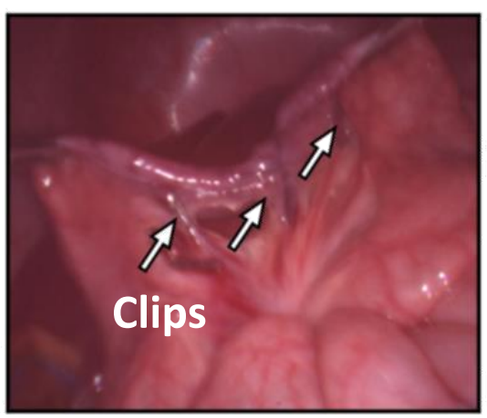
- Forward process: given parameters $x_1, x_2 \sim \mathcal{N}(0,1)$, observation y arises according to
$$y = x_1 + x_2 = g(x_1, x_2)$$
- Inverse $(x_1, x_2) = g^{-1}(\hat{y})$ for given observation $\hat{y}$ is undefined
  - Classical regularization: minimum norm solution $x_1 = x_2 = \frac{\hat{y}}{2}$ (disregards ambiguity!)
- Bayesian solution:
  - Introduce latent variable $z = x_1 - x_2 \sim \mathcal{N}(0,2)$
  - Reparametrize $p(x_1, x_2 \mid \hat{y})$ as $(x_1, x_2) = g_{\text{aug}}^{-1}(\hat{y}, z) = \left(\frac{\hat{y}+z^{(t)}}{2}, \frac{\hat{y}-z^{(t)}}{2}\right)$
    - For $t \in 1, \dots, T$:
      - Sample $z^{(t)} \sim \mathcal{N}(0,2)$ and compute $x_1^{(t)} = \frac{\hat{y}+z^{(t)}}{2}$ and $x_2^{(t)} = \frac{\hat{y}-z^{(t)}}{2}$
    - Return $\left\{\left(x_1^{(t)}, x_2^{(t)}\right)\right\}_{t=1}^{T}$ as a sample from the Bayesian posterior $p(x_1, x_2 \mid \hat{y})$

Generalize this to complex settings (non-linear g, noise, high dimensions) by INNs.

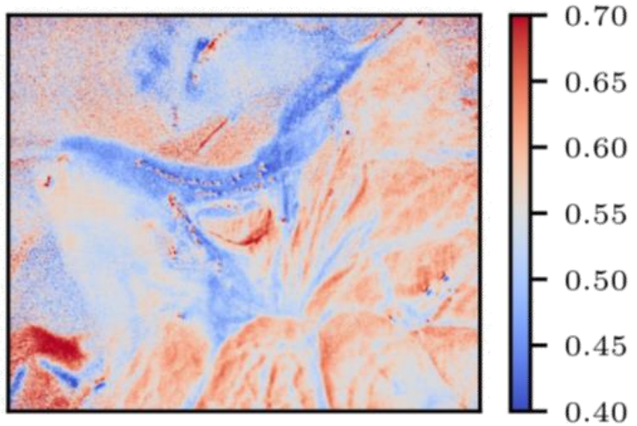
Application: Multispectral Endoscopy

Endoscopes for minimally invasive surgery

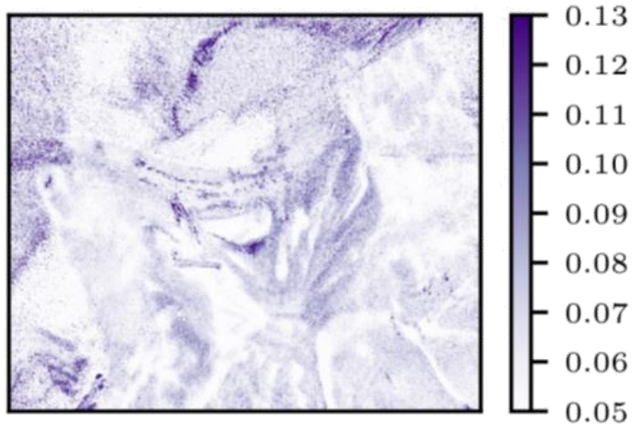
- can be equipped with a multispectral camera
- tissue state x (e.g. blood oxygenation) affects the observed spectrum y
- **Task:** given spectrum, find posterior distribution of tissue state parameters
- Forward process $s(x)$ is implemented by Monte Carlo simulation



c) RGB image



a) Median s_{O_2}

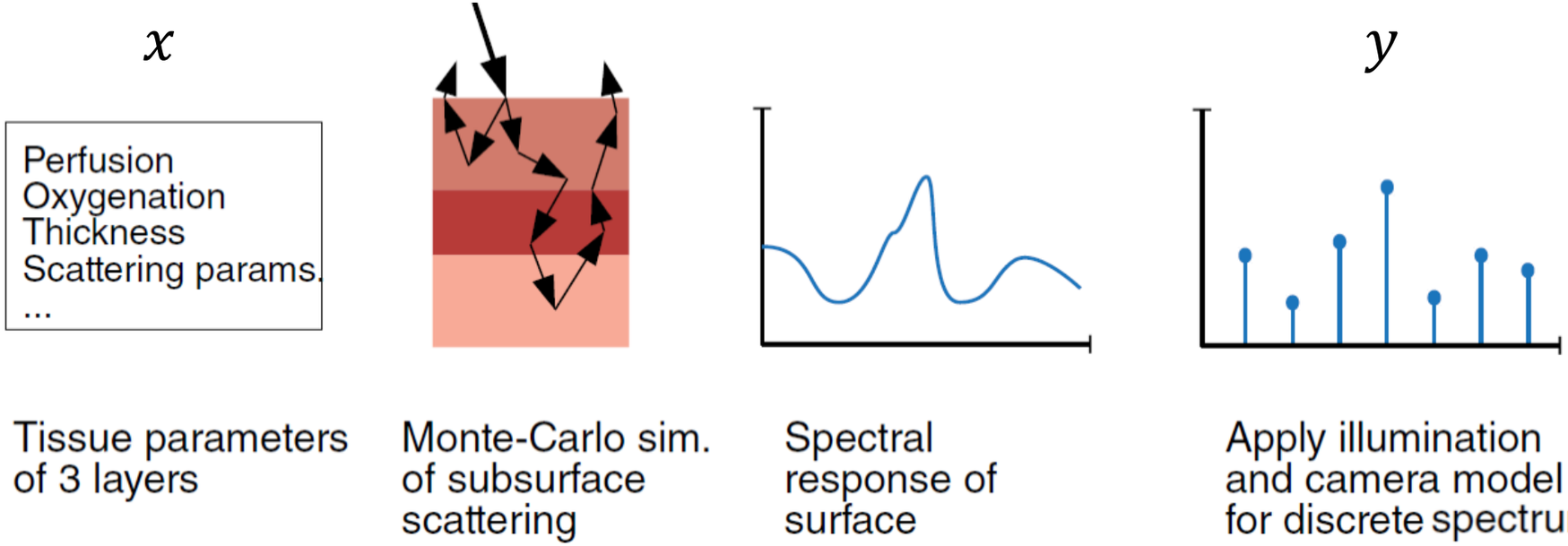


b) Est. uncertainty

Application: Multispectral Endoscopy

Invert the forward process $s(x)$ implemented by Monte Carlo simulation:

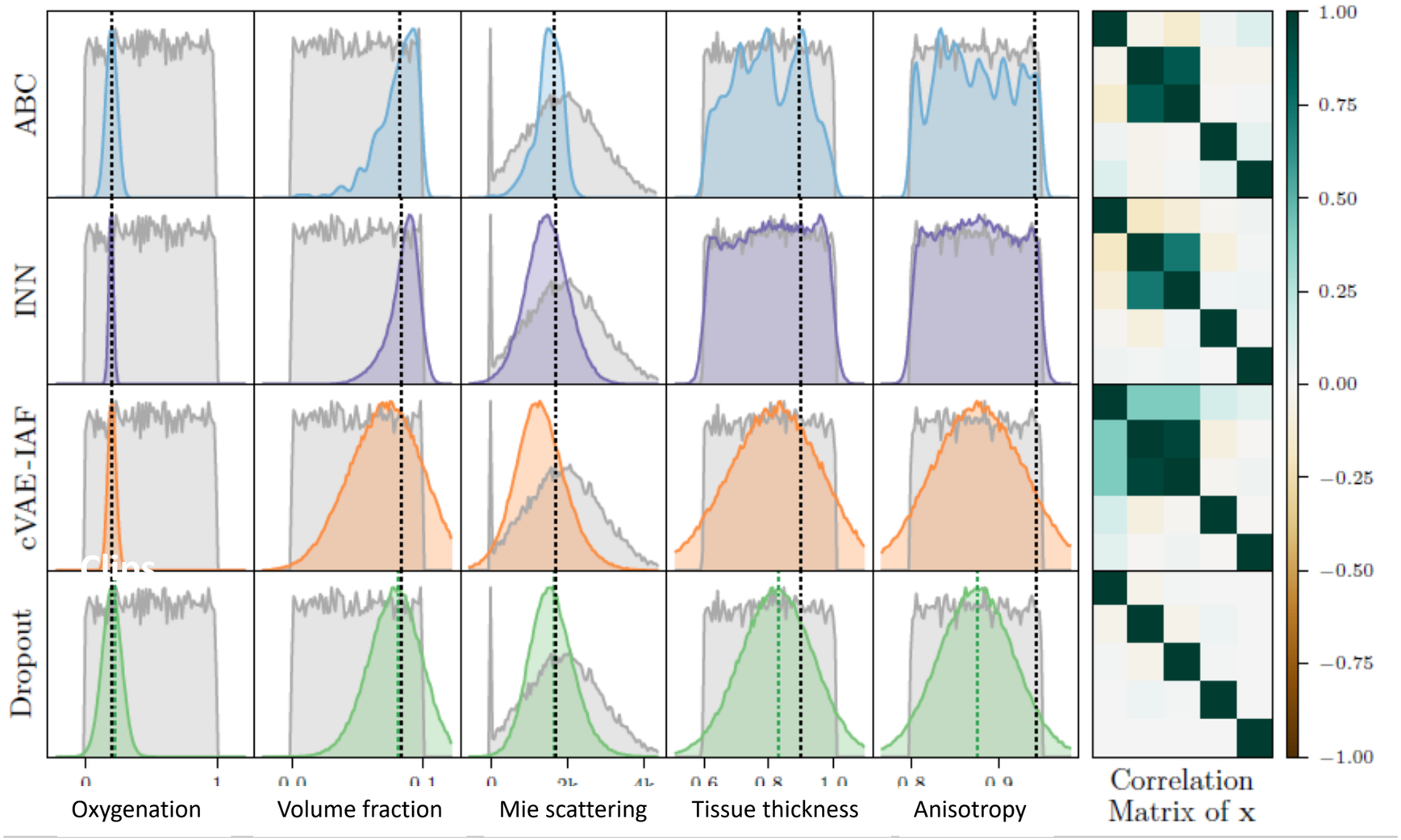
- training: INN learns $[y, z] = f_\theta(x) \approx s_{aug}(x)$ with $p(z) \sim \mathcal{N}(0, \mathbb{I})$
- inference: given observed spectrum $\hat{y}$, sample $\{z_i \sim p(z)\}_{i=1}^{M}$ and compute posterior sample $\left\{x_i = f_\theta^{-1}(\hat{y}, z_i)\right\}_{i=1}^{M}$ (independently for every pixel)
- determine mean and variance from $\{x_i\}$ – works especially well for blood oxygenation



Application: Multispectral Endoscopy

Results

- INN performs well
- not all parameters are identifiable

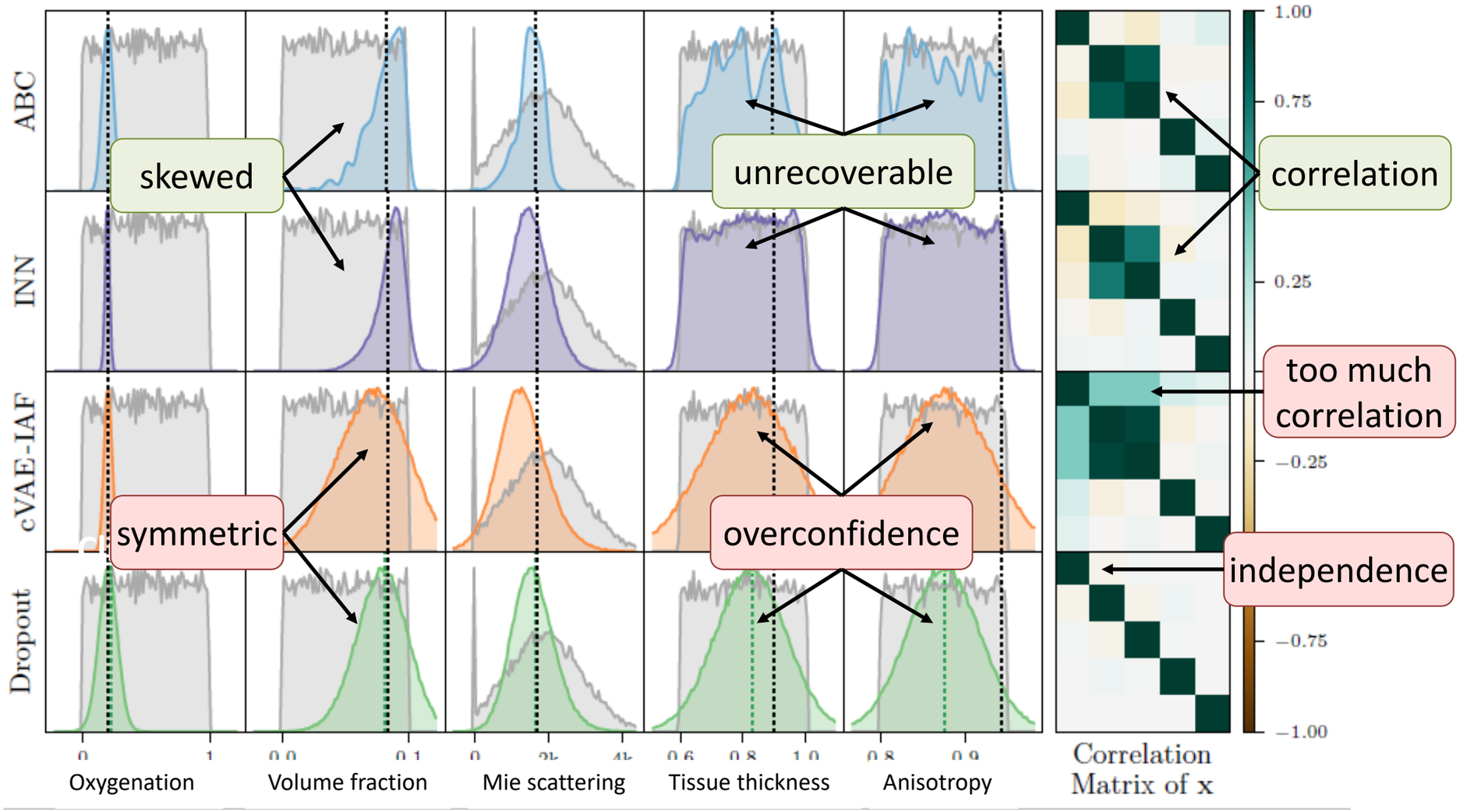


Application: Multispectral Endoscopy

Results

- INN performs well
- not all parameters are identifiable

- incorrect results for other methods
  - skewed distribut. appear symmetric
  - non-identifiable parameters have spurious mode
  - correlation is too weak or too strong

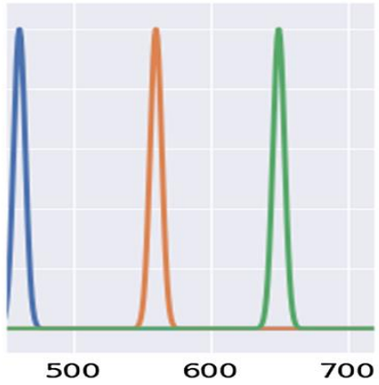


Experimental Design for Multispectral Endoscopy

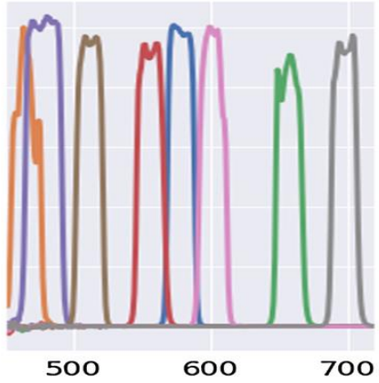
Analysis of posteriors: Which camera should be used?

- 3 to 27 spectral channels
- Which gives reliable results at best price and usability?

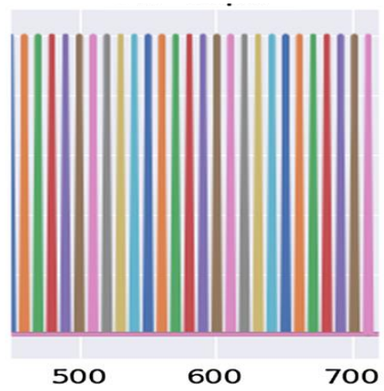
3 spectral channels



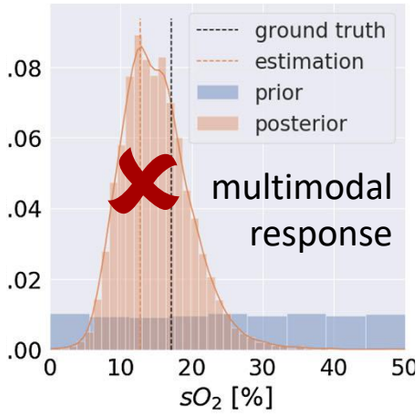
8 spectral channels

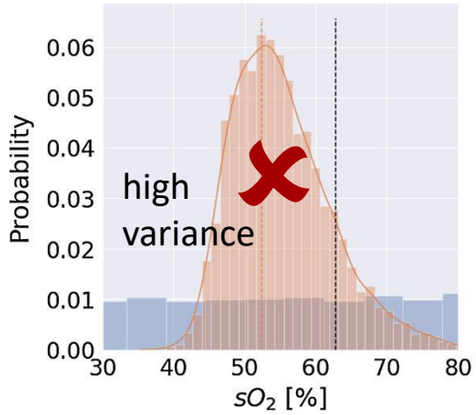


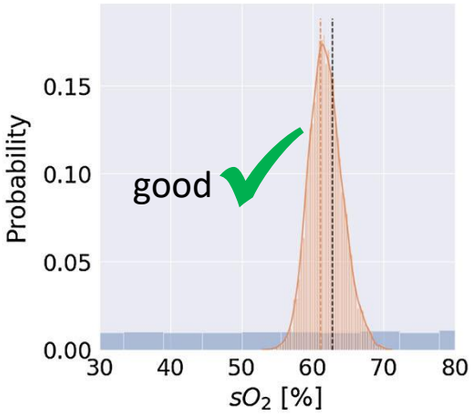
27 spectral channels



- posterior oxygen level histograms:







⇨ camera with 8 channels offers best trade-off between price and accuracy

INN Architecture for Endoscopy Application

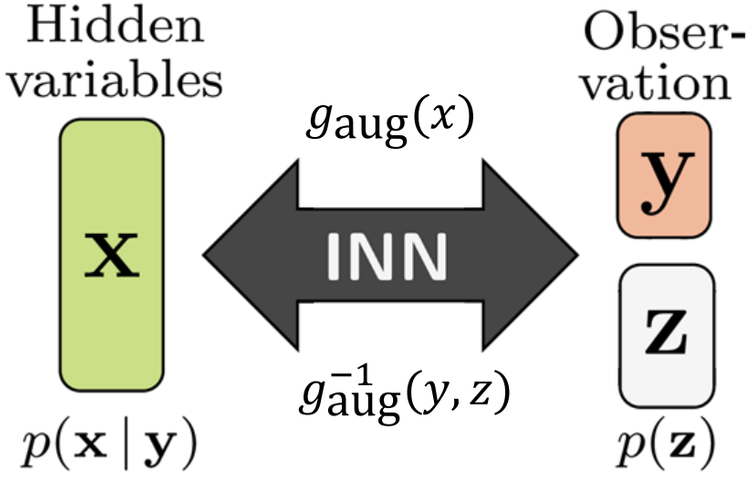
- Forward process: given tissue parameters x, spectrum y arises from MC simulation g

$$y = g(x)$$

- Bayesian solution:
  - Introduce latent variables z collecting the information about x that got lost in $y = g(x)$

$$y, z = g_{\text{aug}}(x)$$

  - Train INN for $g_{\text{aug}}(x)$ with $p(z) = \mathcal{N}(0, \mathbb{I})$ and $y \perp z$, using synthetic training data from the simulation
  - Inference for real observation y_{obs}:
    - For $t \in 1, \ldots, T$:
      - Sample $z^{(t)} \sim \mathcal{N}(0, \mathbb{I})$
      - compute $= x^{(t)} = g_{\text{aug}}^{-1}\left(y_{obs}, z^{(t)}\right)$
    - Return $\left\{\left(x^{(t)}\right)\right\}_{t=1}^{T}$ as a sample from Bayesian posterior $p(x \mid y_{obs})$

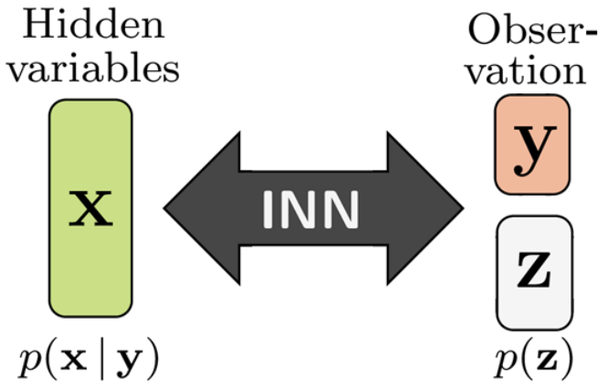


INN Architectures for Inverse Bayesian Inference

Augmented latent space

training: $(y, z) = f_\theta(x)$
s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$
compute $x = f_\theta^{-1}(\hat{y}, z)$
⇨ $x \sim p(x \mid \hat{y})$

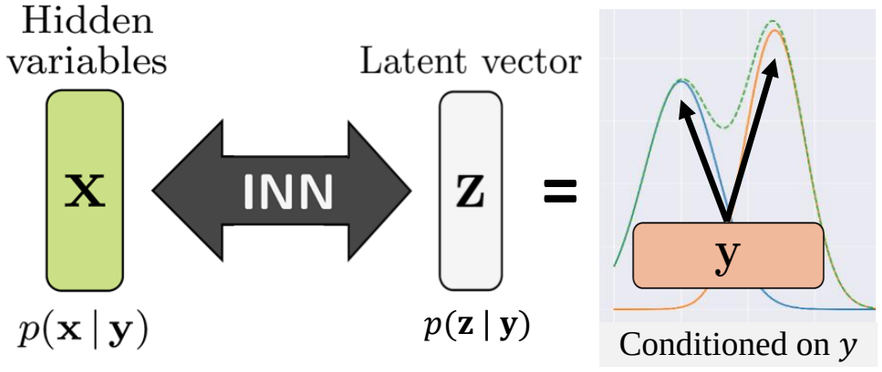


the original

Latent mixture INN

training: $z = f_\theta(x)$
s.t. $p(z) = \mathrm{GMM}(z; y) = \sum_y \mathcal{N}(\mu_y, \Sigma_y)$

inference: sample $z \sim \mathcal{N}(\mu_{\hat{y}}, \Sigma_{\hat{y}})$
compute $x = f_\theta^{-1}(z)$
⇨ $x \sim p(x \mid \hat{y})$

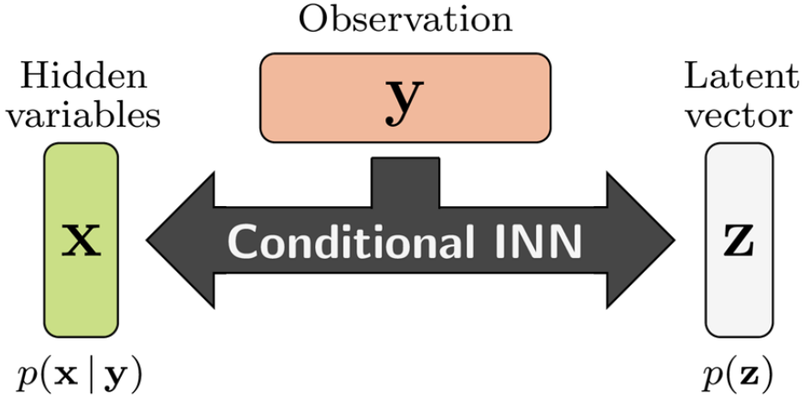


disentanglement

Conditional INN

training: $z = f_\theta(x; y)$
s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

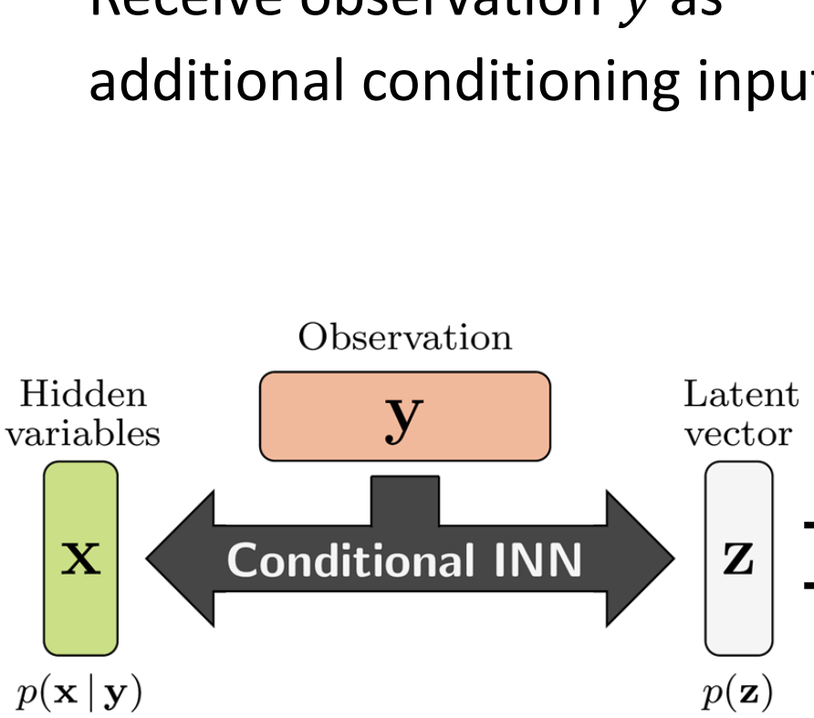
inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$
compute $x = f_\theta^{-1}(z; \hat{y})$
⇨ $x \sim p(x \mid \hat{y})$



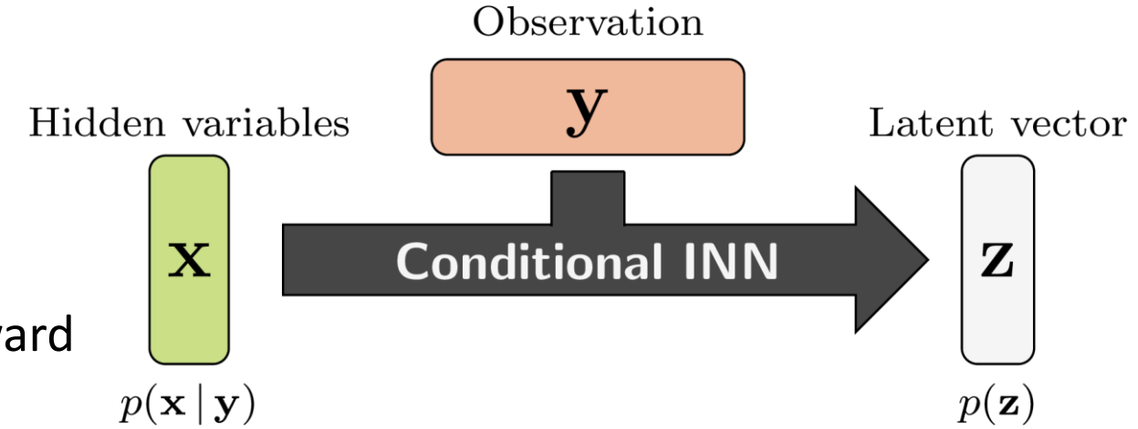
simulation-based inference

Conditional INN (cINN)

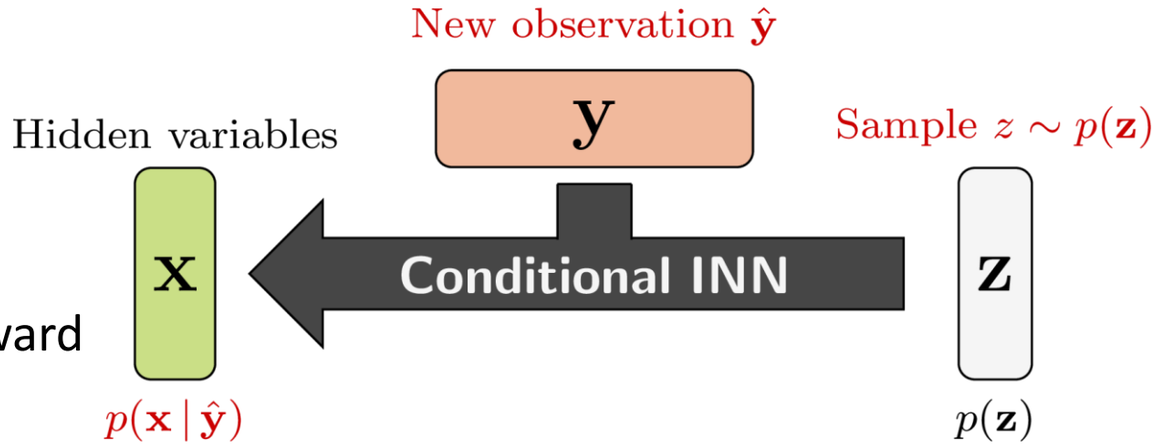
Receive observation y as additional conditioning input



Training:
run cINN forward



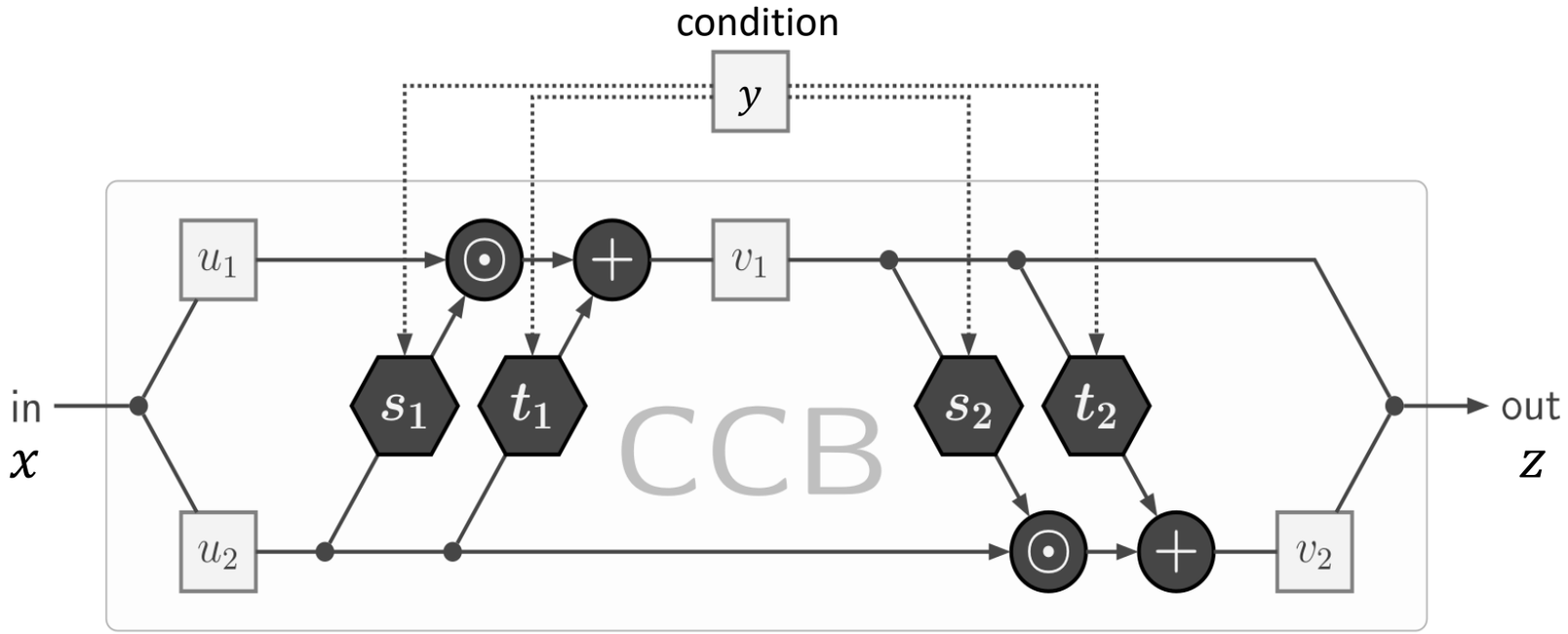
Inference:
run cINN backward



cINN Architecture

Conditional coupling layers: simple generalization of standard coupling layers

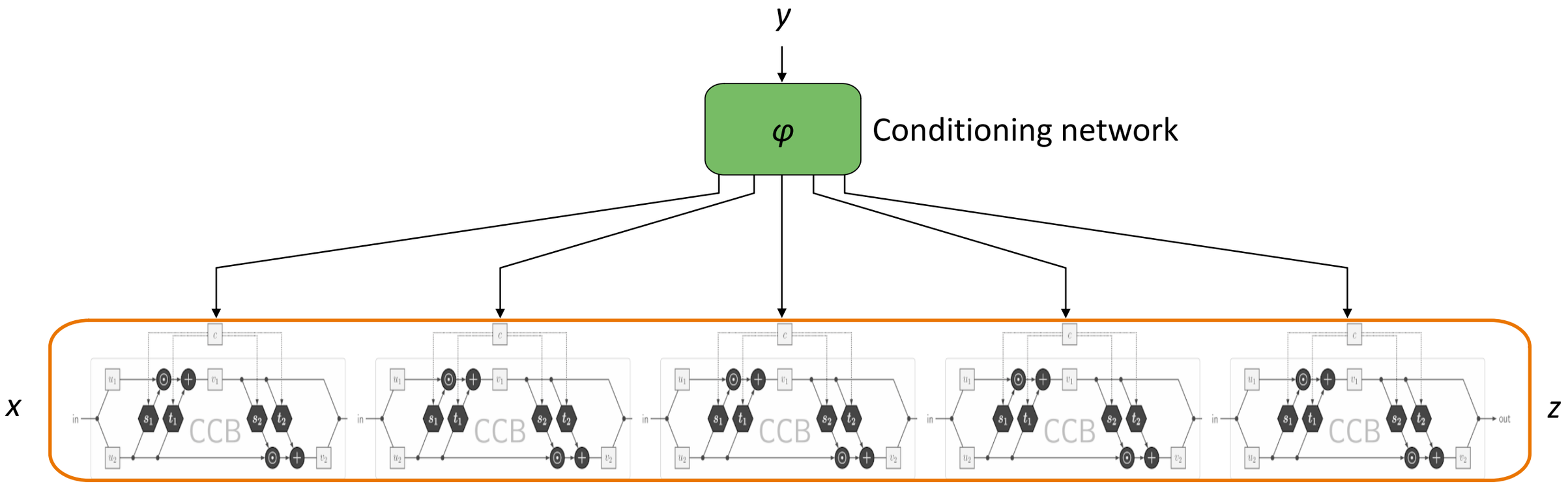
- Coefficient networks $s_{1,2}, t_{1,2}$ get additional conditioning input y
- For fixed condition, coupling blocks are still invertible (since coefficient networks are never inverted)
- Computes $\boldsymbol{z} = f(\mathbf{x}; \mathbf{y})$ and its inverse $\mathbf{x} = f^{-1}(\boldsymbol{z}; \mathbf{y})$



cINN Architecture

Conditional INN: sequence of conditional coupling layers

- If condition is complex (e.g. image): preprocessing via conditioning network $\phi(\mathbf{y})$ (= feature extraction)
- Computes $\boldsymbol{z} = f(\mathbf{x}; \boldsymbol{\phi}(\mathbf{y}))$ and its inverse $\mathbf{x} = f^{-1}(\boldsymbol{z}; \phi(\mathbf{y}))$



cINN Training

Estimated conditional density $\hat{p}(x \mid y)$ expressed via 'reparameterization trick'

- Let the cINN represent the function $z = f_\theta(x\,; y)$ (now also dependent on y)
- Train cINN such that $p_z(z) \approx \mathcal{N}(0, \mathbb{I})$
- Then $\hat{p}(x \mid y) \approx p^*(x \mid y)$ is defined by the change-of-variables formula

$$\hat{p}(x \mid y) = p_z\big(z = f_\theta(x\,; y)\big) \left| \det\left(\frac{\partial f_\theta(x\,; y)}{\partial x}\right)\right|$$

Can be trained with maximum likelihood loss as before:

$$\hat{\theta} = \arg\min_\theta \sum_{i \in \mathcal{TS}} -\log \hat{p}\big(x^{(i)} \,\big|\, y^{(i)}\big)$$

$$= \arg\min_\theta \sum_{i \in \mathcal{TS}} \left[\left\| f_\theta\big(x^{(i)}; y^{(i)}\big)\right\|_2^2 - \sum_{l=1}^{L} \tilde{s}_l \left(x_l^{(i)}; y^{(i)}\right)\right]$$

Since f_θ is invertible (given $\hat{y}$), we get a generative model for free:

$$x \sim \hat{p}(x \mid \hat{y}) \quad \Leftrightarrow \quad z \sim \mathcal{N}(0, \mathbb{I}) \quad \text{and} \quad x = f_{\hat{\theta}}^{-1}(z; \hat{y})$$

BayesFlow: cINNs for Simulation-Based Inference

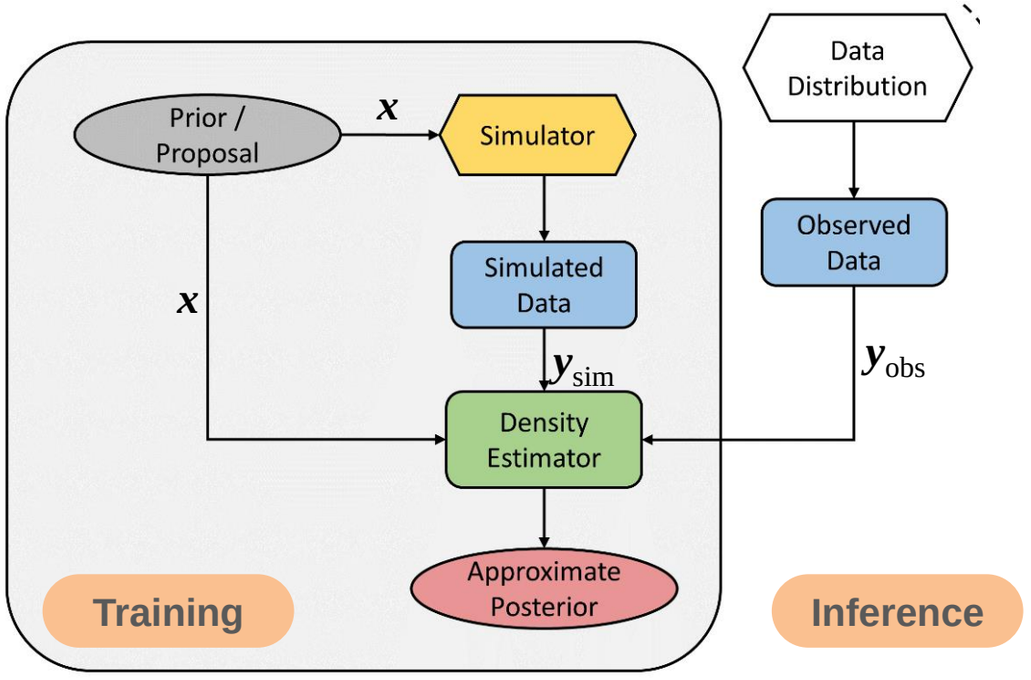
- Train an approximate posterior $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}) \approx p(\boldsymbol{x} \mid \boldsymbol{y})$ that works for any $\hat{\boldsymbol{y}} = \boldsymbol{y}_{\text{obs}}$
  - Expensive one-time (upfront) training of $\hat{p}$ using *simulated* data
  - Each inference query $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$ (for different $\boldsymbol{y}_{\text{obs}}$) is then cheap

  ⇨ Training effort quickly amortizes over multiple cheap queries

- Can be elegantly realized with cINNs: **BayesFlow**
  - Meta-algorithm:
    - Repeat: (training phase)
      - Simulate parameters $\boldsymbol{x} \sim p(\boldsymbol{x})$
      - Run the simulation $\boldsymbol{y}_{\text{sim}} = g(\boldsymbol{x};\ \boldsymbol{\xi})$
      - Perform maximum likelihood training of a cINN with (a batch of) $(\boldsymbol{x}, \boldsymbol{y}_{\text{sim}})$ pairs

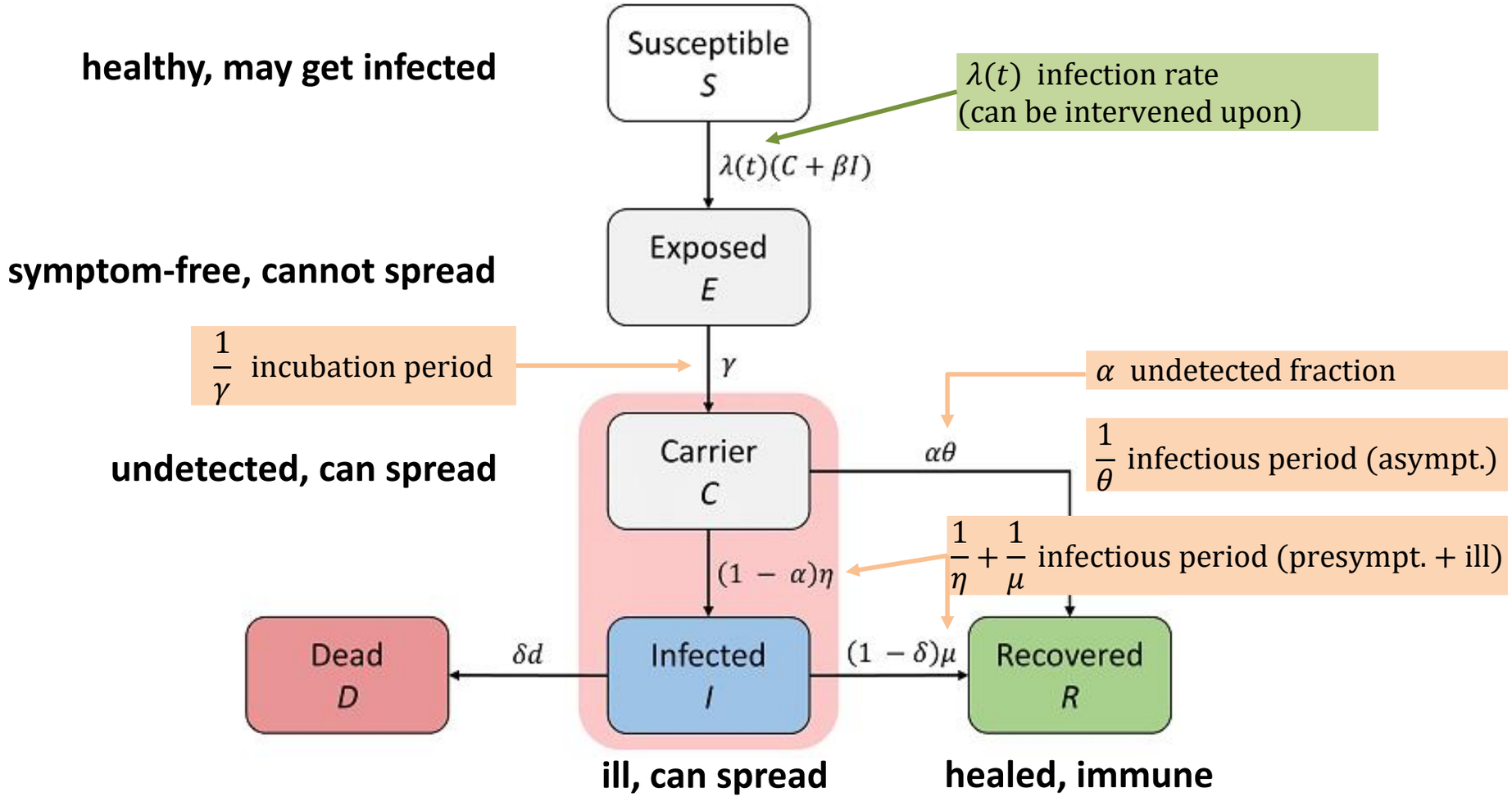
      Until convergence to $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{sim}})$

    - For each observation $\boldsymbol{y}_{\text{obs}}$: (inference phase)
      - Use cINN to compute $\hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})$ or create sample $\{\boldsymbol{x}_i \sim \hat{p}(\boldsymbol{x} \mid \boldsymbol{y}_{\text{obs}})\}$



BayesFlow for Epidemiology: Covid-19 Forward Model

- Forward model: SECIRD compartmental model (Lotka-Volterra type ODE system):



BayesFlow for Epidemiology: Covid-19 Forward Model

- Forward model: Lotka-Volterra type ODE system, e.g. SECIRD:

healthy, may get infected

$$\frac{dS}{dt} = -\lambda(t)\left(\frac{C+\beta I}{N}\right)S$$

symptom-free, cannot spread

$$\frac{dE}{dt} = \lambda(t)\left(\frac{C+\beta I}{N}\right)S - \gamma E$$

undetected, can spread

$$\frac{dC}{dt} = \gamma E - (1-\alpha)\eta C - \alpha\theta C$$

ill, can spread

$$\frac{dI}{dt} = (1-\alpha)\eta C - (1-\delta)\mu I - \delta d I$$

healed, immune

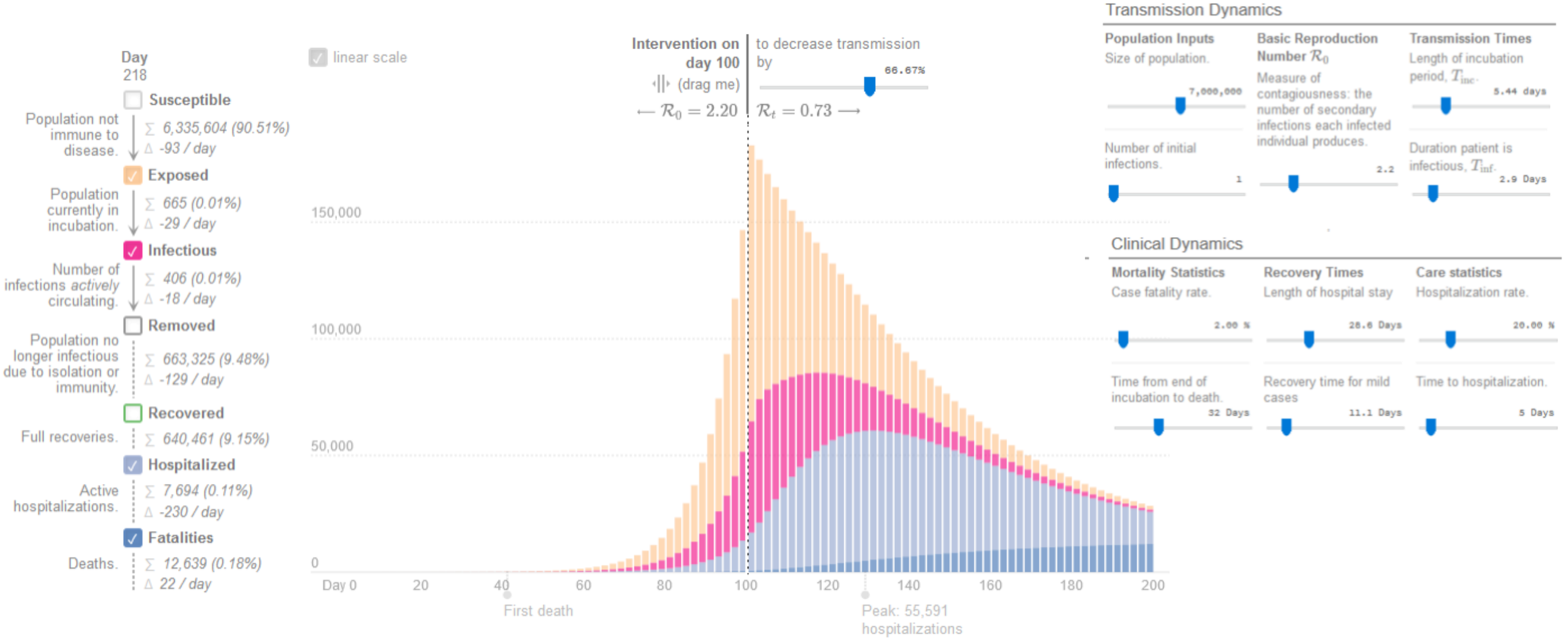
$$\frac{dR}{dt} = \alpha\theta C + (1-\delta)\mu I$$

dead

$$\frac{dD}{dt} = \delta d I$$

Radev et al. **"Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany"**, arXiv 2020.

Forward Model: Epidemic Calculator



Goh: **"Epidemic Calculator"**, https://gabgoh.github.io/COVID/, 2020.

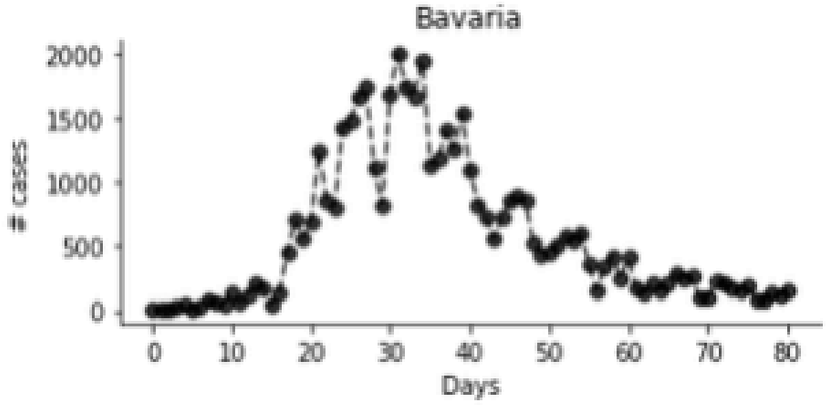
BayesFlow for Epidemiology: Enhanced Covid-19 Forward Model

- Enhance realism
  - Observation model: reporting delay, noise, weekly modulation

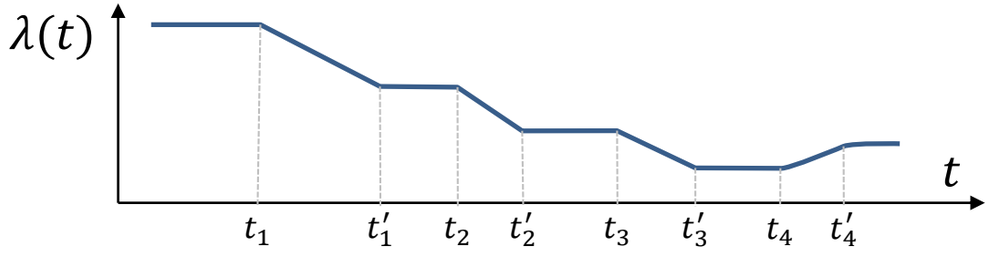
$$I_t^{(obs)} = I_{t-1}^{(obs)} + (1 - f_I(t))\,(1-\alpha)\,\eta\,C_{t-D_I} + \sqrt{I_{t-1}^{(obs)}}\,\sigma_I\,\xi_t$$

$$R_t^{(obs)} = R_{t-1}^{(obs)} + (1 - f_R(t))\,(1-\delta)\,\mu\,I_{t-D_R} + \sqrt{R_{t-1}^{(obs)}}\,\sigma_R\,\xi_t$$

$$D_t^{(obs)} = D_{t-1}^{(obs)} + (1 - f_D(t))\,\delta\,d\,I_{t-D_D} + \sqrt{D_{t-1}^{(obs)}}\,\sigma_D\,\xi_t$$



  - Intervention model:
    four intervals where countermeasures
    were implemented or relaxed



  - Total: 34 parameters with uninformative or very wide priors
  - Forward simulation is easy: sample from prior, solve ODEs by Runge-Kutta
  - **Inverse: find parameter posteriors from observed time series of detected, dead, recovered cases.**

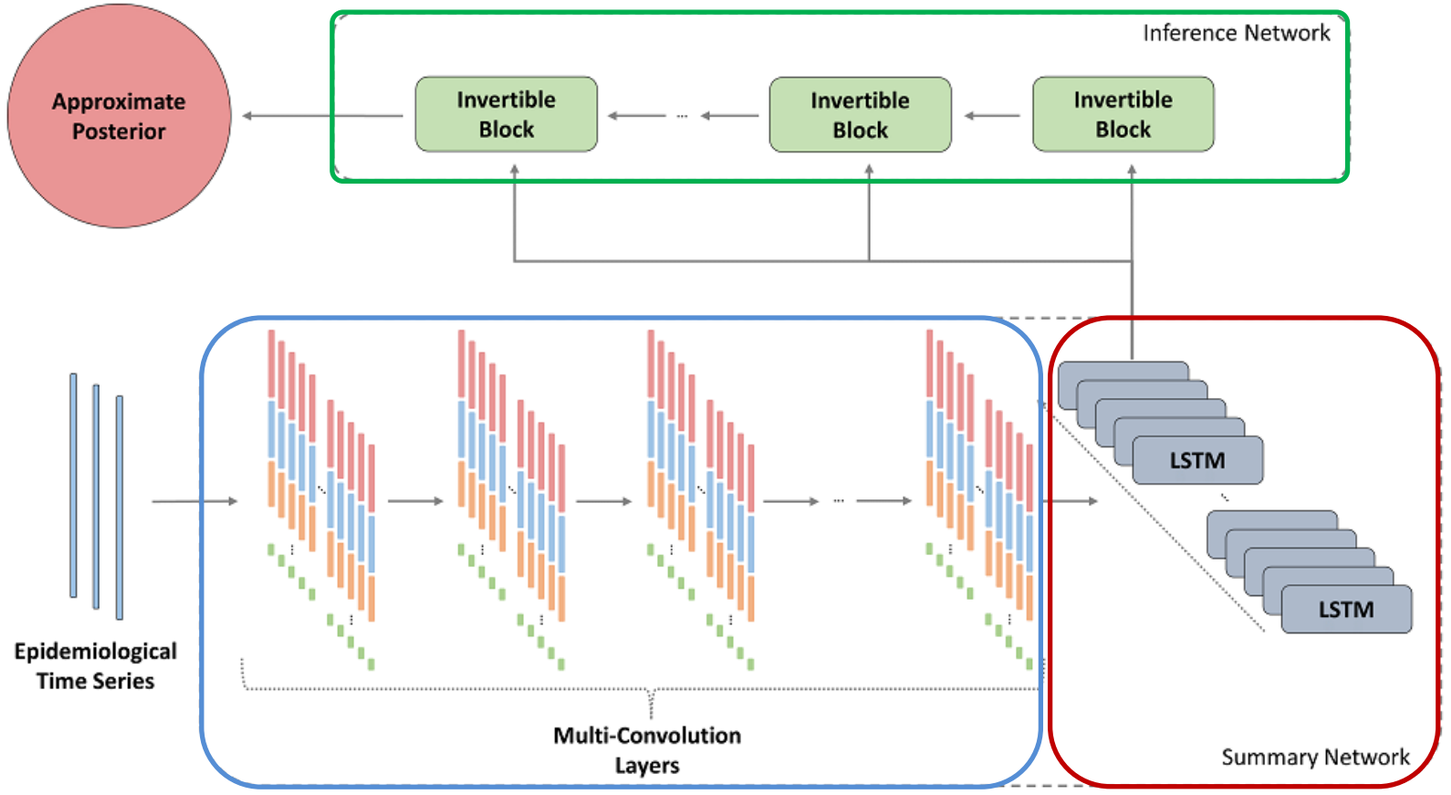
BayesFlow for Epidemiology: The Networks

- Inference problem: observation sequence (IRD) ⇨ parameter posteriors
  - Solve with BayesFlow network: cINN with statistical preprocessing networks for y

Convolutional:
- noise reduction
  feature detection

Recurrent (LSTM):
- variable-length sequence
  to fixed size summary

Invertible (cINN):
- posterior inference

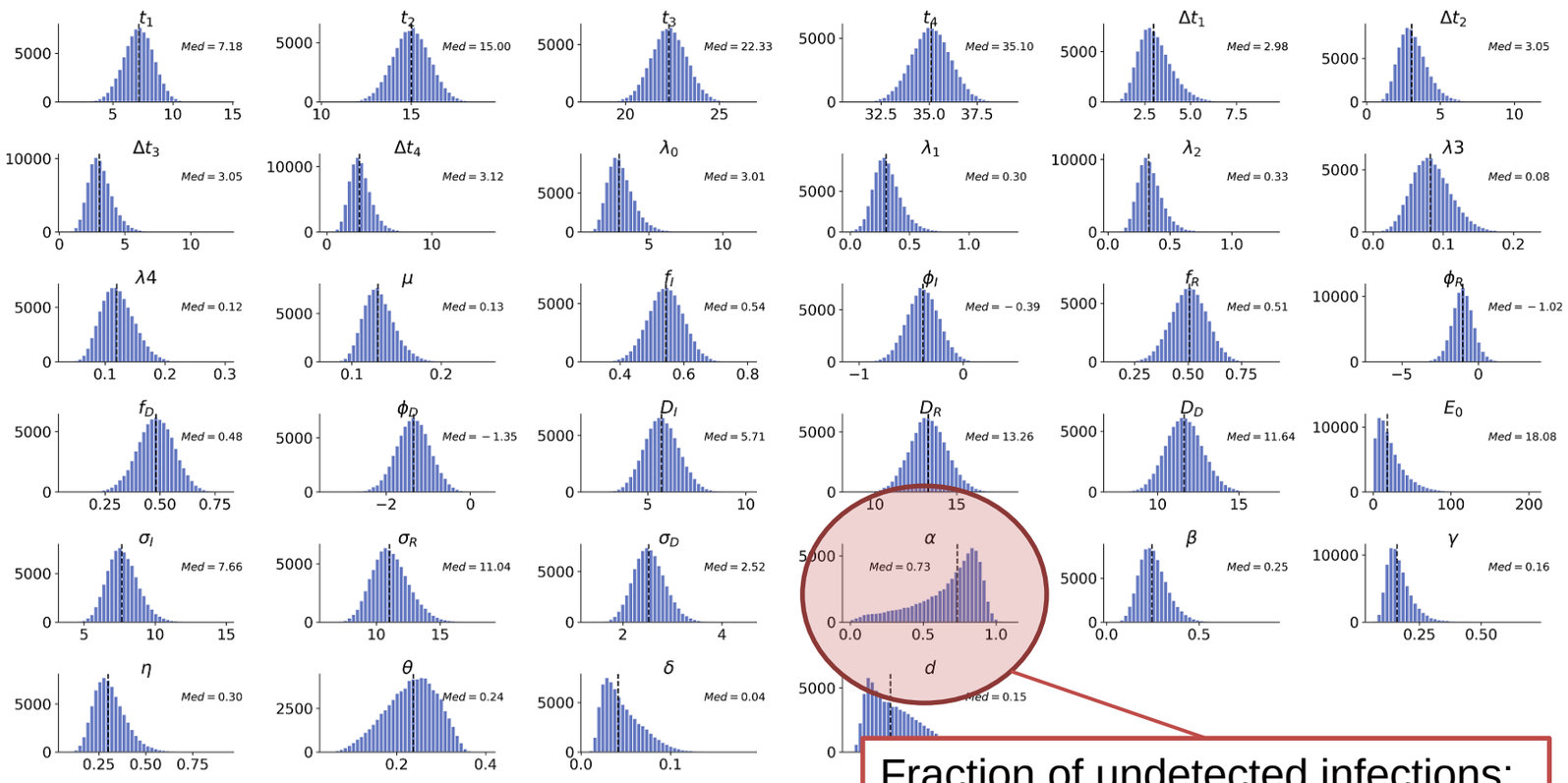


  - Training: end-to-end optimization of maximum likelihood loss with 50000-75000 simulations

BayesFlow for Epidemiology: Covid-19 Marginal Posteriors

Results: marginal posteriors for first wave in Germany (March – June 2020, 81 time steps)

- Fraction of infections remaining undetected:
  66% (median), 75% (mode)
- Serial interval: 9-10 days
- High likelihood to transmit disease *before* diagnosis
- time to recovery:
  4.6 days (undetected infections)
  11.3 days (diagnosed cases)
  (3.2 + 8.1 days before/after diagnosis)
- often non-Gaussian behavior



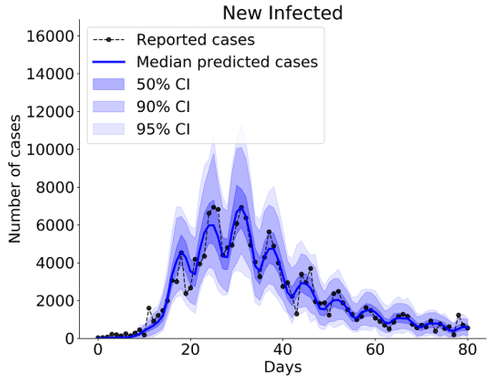
Fraction of undetected infections:
uniform prior ⇨ peaked posterior

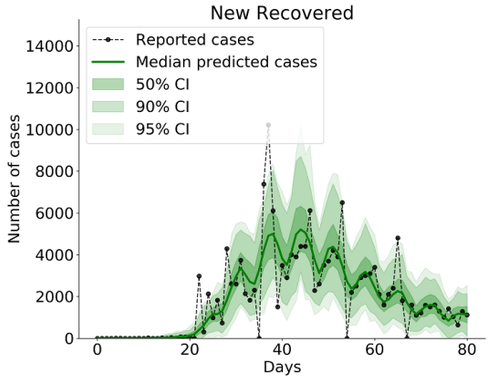
Correspond well to clinical findings

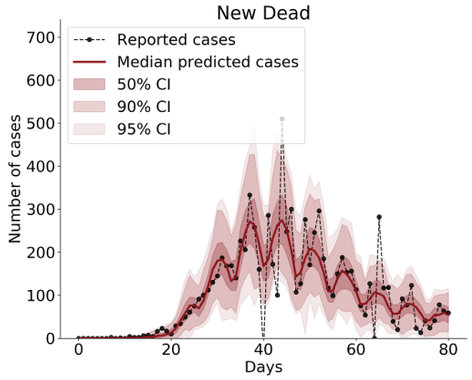
BayesFlow for Epidemiology: Covid-19 Predictive Posteriors

- Results: predictive posteriors for first wave in Germany (March – June 2020, 81 time steps)

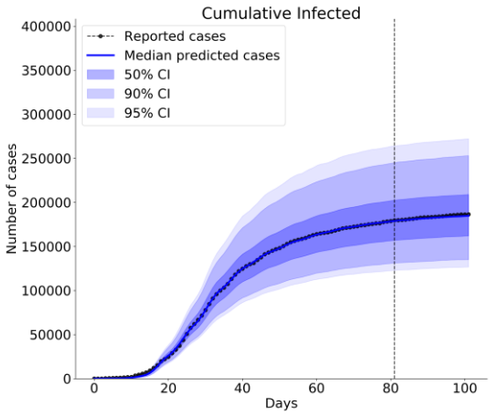
daily:

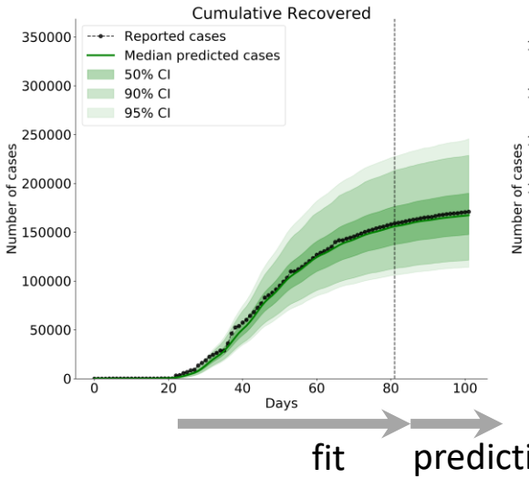


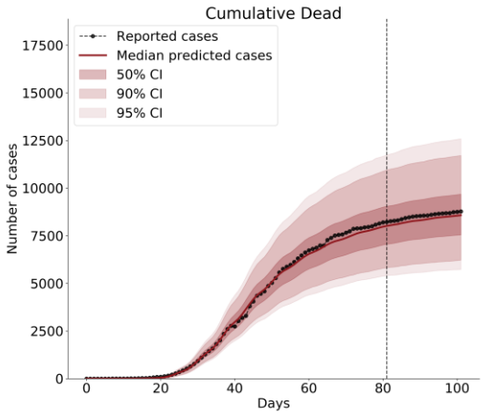




cumulated:

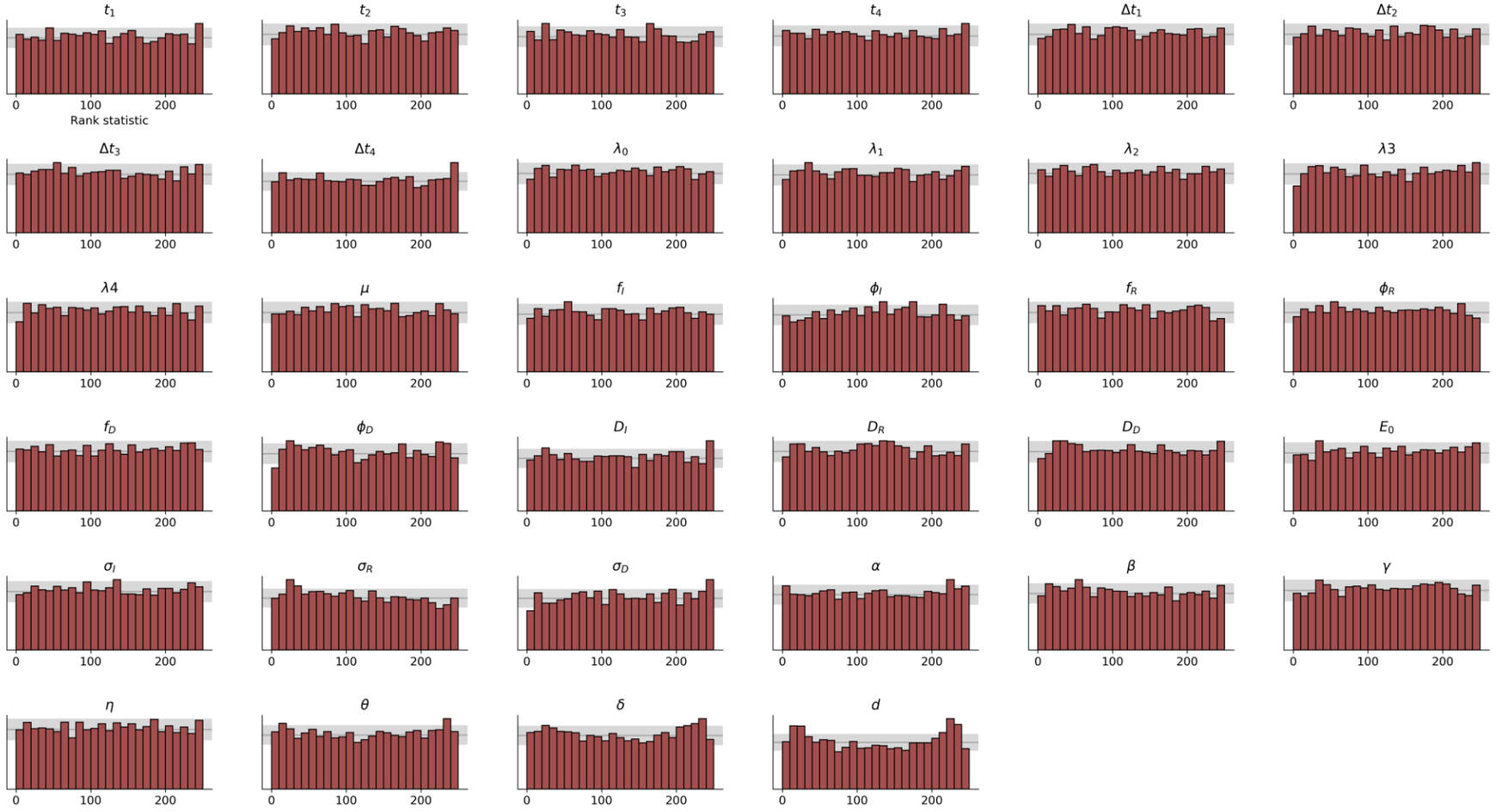






BayesFlow for Epidemiology: Covid-19 Uncertainty Calibration

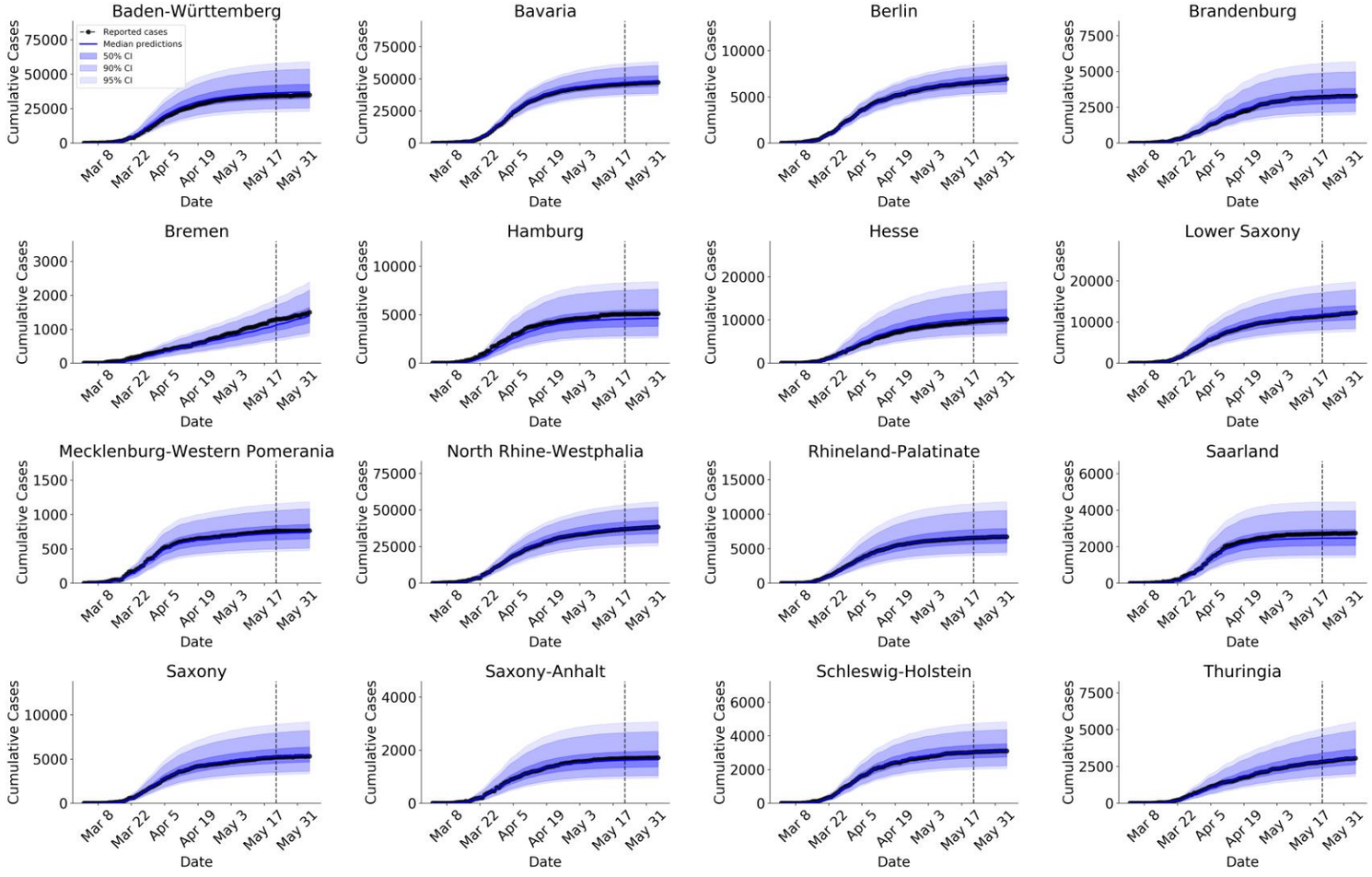
- Well-calibrated uncertainty quantification



Radev et al. **"Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany"**, arXiv 2020.

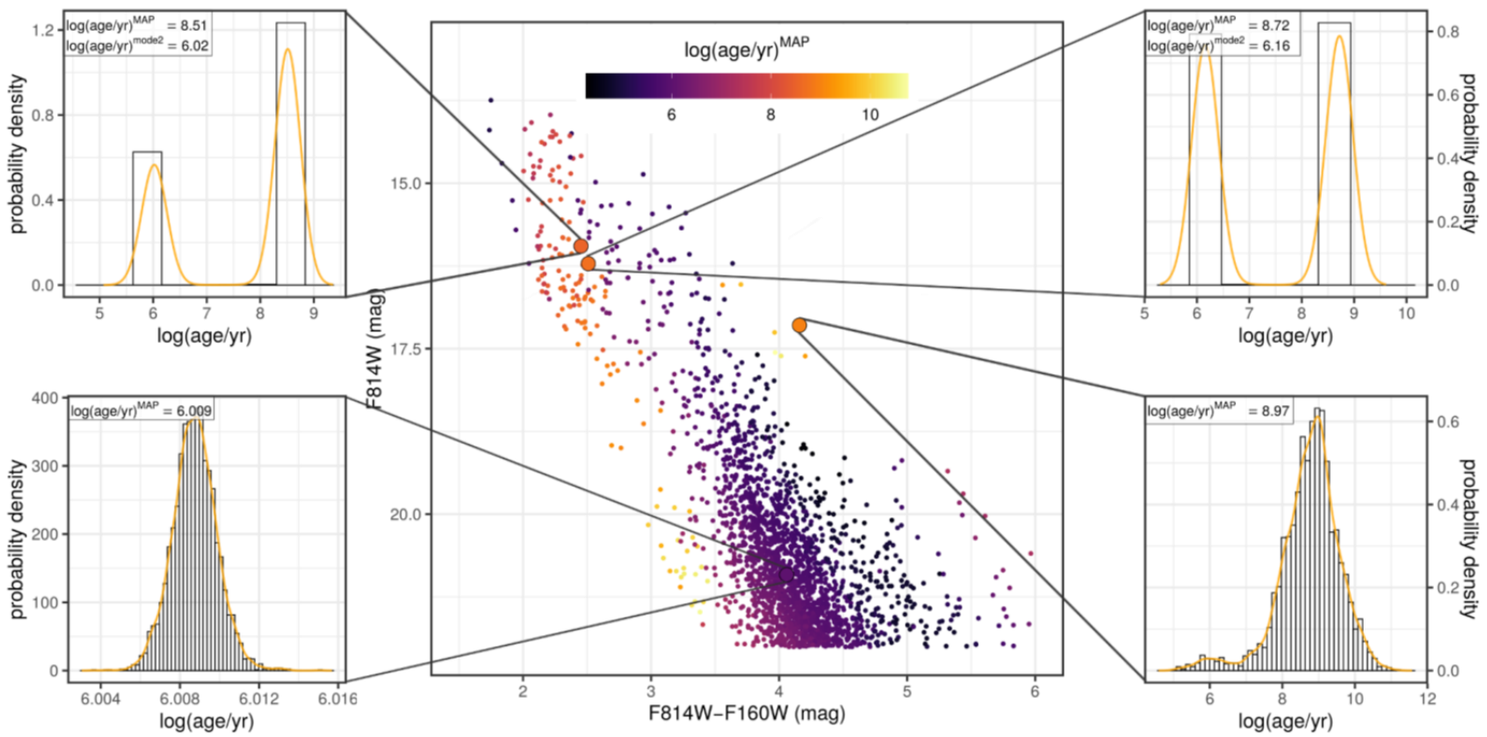
BayesFlow for Epidemiology: Covid-19 Amortized Inference

- Amortized inference: same model works for German states (no re-training)



Application: Stellar Parameters Prediction from Photometry

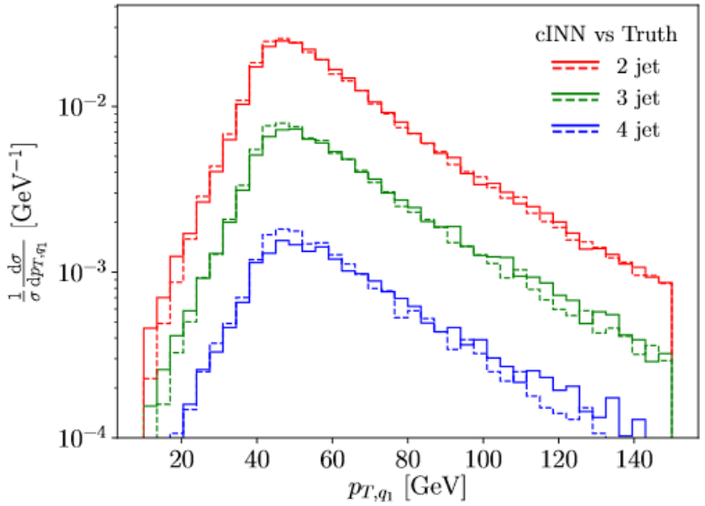
- Training with Stellar Evolution Model – PARSEC (Bressan et. al 2012)
  - Forward problem: simulate spectral properties from stellar parameters (mass, age, metallicity,…)
  - Inverse problem: infer stellar parameters from Hubble space telescope observations
- Results on Westerlund 2 young star forming cluster
  - Observations: color magnitude diagrams
  - Parameters: age, initial and current mass, luminosity, surface gravity, effective temperature
  - Well calibrated posteriors
  - MAP estimates close to truth
  - Physically plausible ambiguities: same spectral signal for young/heavy and older/lighter starts



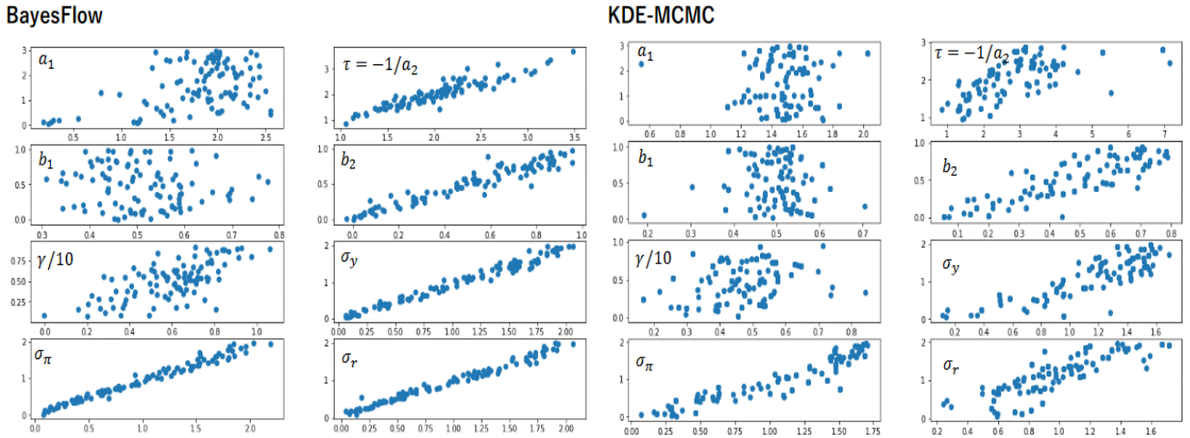
More Application Examples

- ZW-production unfolding at the LHC
  - can handle variable number of jets:

$$pp \to ZW^{\pm} + \text{jets} \to (\ell^{-}\ell^{+})\ (jj) + \text{jets}$$



- Agent-based models in finance / economy



. Bellagente et al. **"Invertible Networks or Partons to Detector and Back Again"**, arXiv, 2020.
Shiono "**Estimation of agent-based models using BayesFlow**", SSRN, 2020.

Application: Image-to-Image Translation

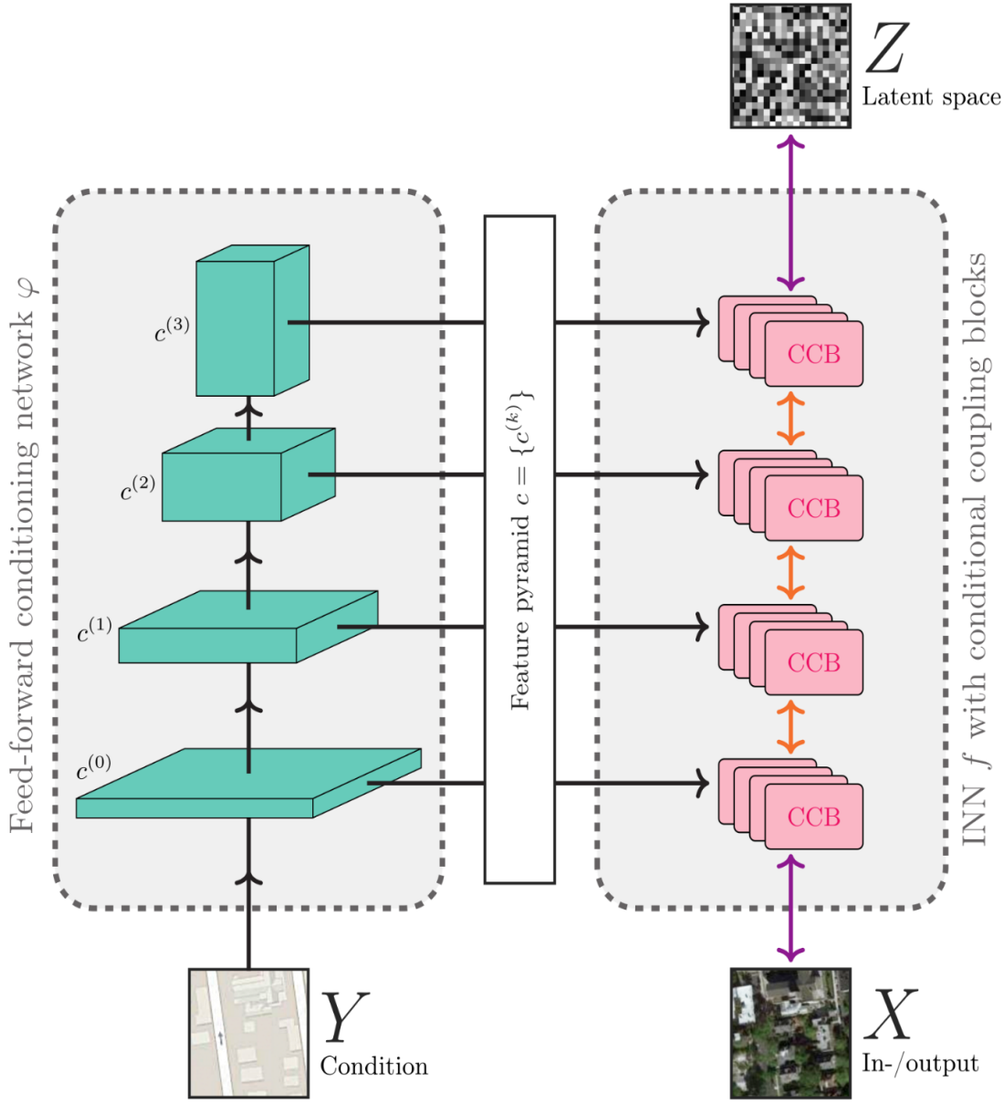
cINN for Image-to-Image Translation

- Example application: transform daylight images to night images
  - Condition y is the daylight image

  ⇨ preprocess with a feature-detection CNN
    - Layers of the CNN compute conditions $c^{(l)}$ at different resolutions / scales

  - Invertible network turns $p(z)$ into $p(x \mid y)$
    - Use multi-resolution features $c^{(l)}$ in the coupling block of corresponding resolution

  ⇨ creates diverse night images x for each y



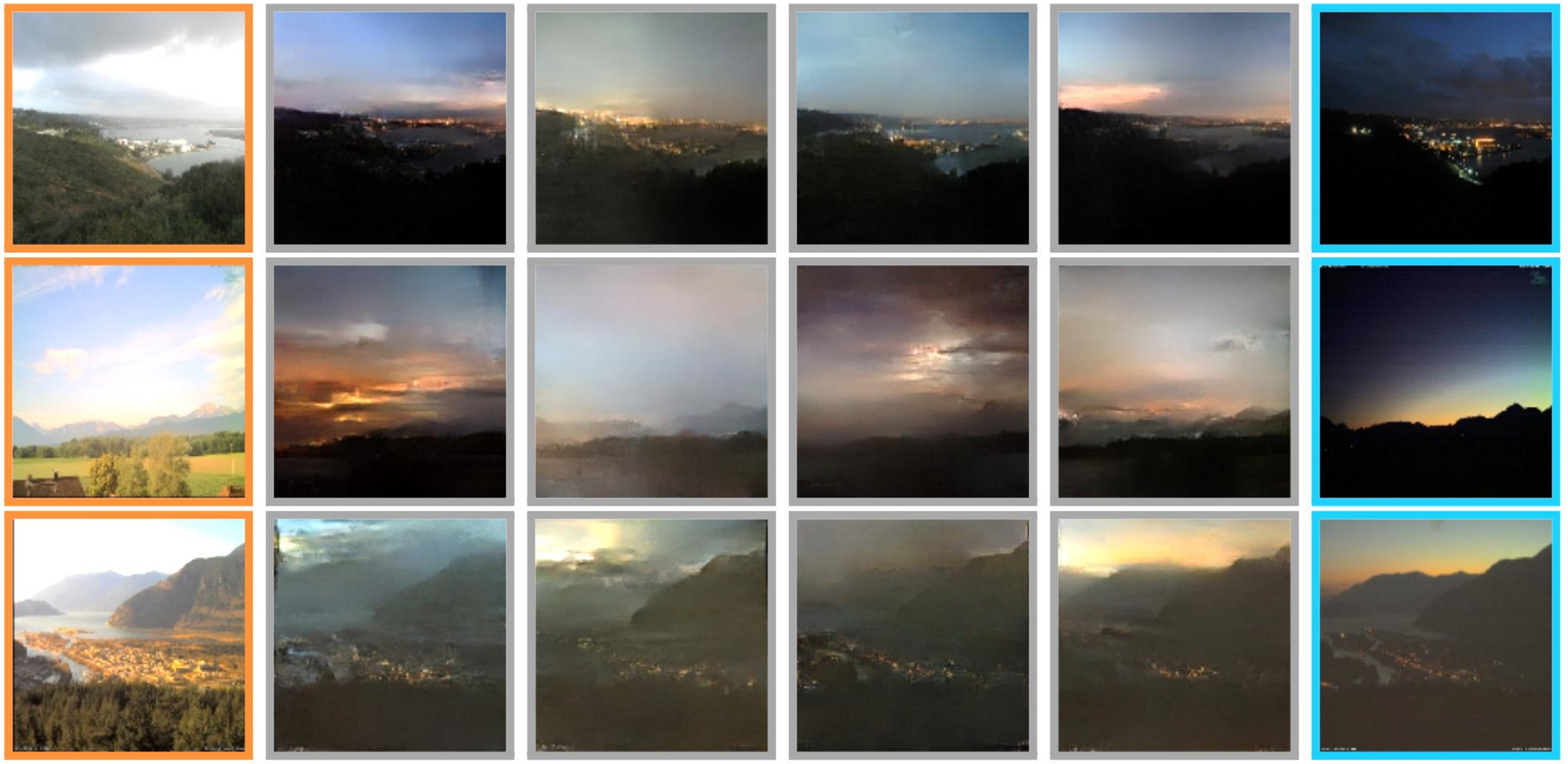
cINN for Image-to-Image Translation

- Results:

Condition *y*
Day image

Generated *x*
Night images

Ground truth *x*
Night image

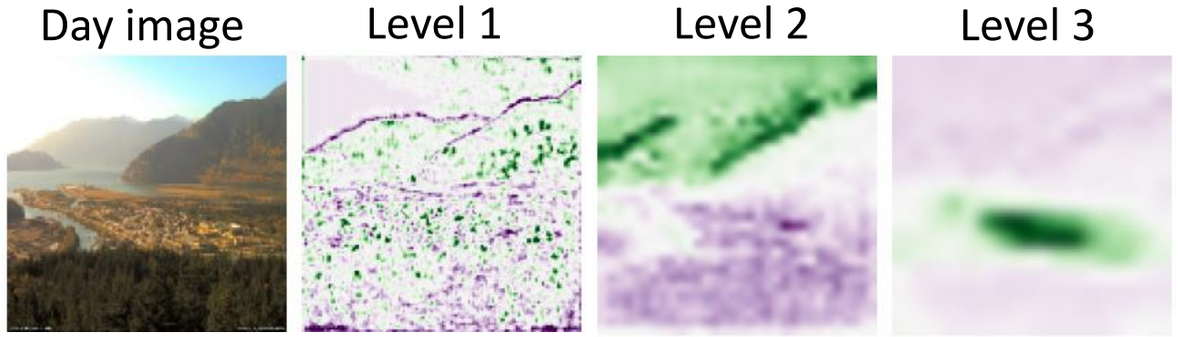


Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

cINN for Image-to-Image Translation

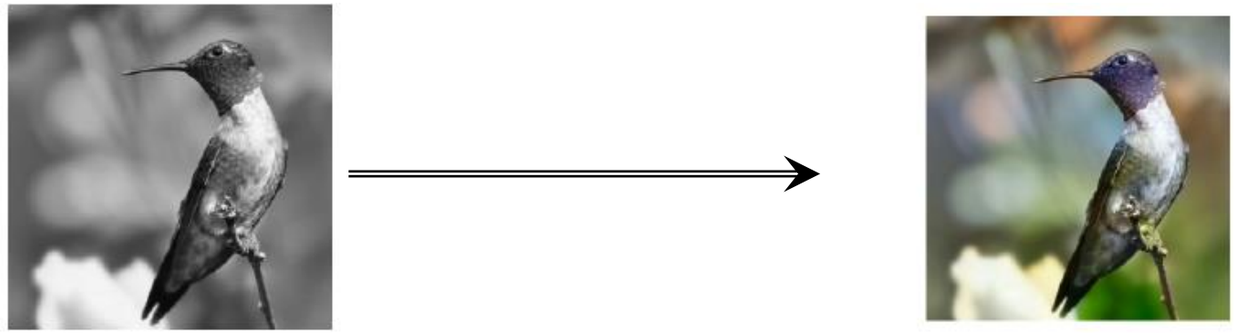
- Results:

- Multi-scale features learned by the conditioning network:
  - Level 1: edges and texture
  - Level 2: foreground / background
  - Level 3: populated areas (lights!)



Application: Colorization

- Inverse problem:
  - given grayscale image y (256×256, L-channel in Lab color space)
  - create **realistic** color channels $\hat{x} = [a, b]$ $(64 \times 64 \times 2)$



y

Application: Colorization

- Inverse problem:
  - given grayscale image y (256×256, L-channel in Lab color space)
  - create **realistic** color channels $\hat{x} = [a, b]$ $(64 \times 64 \times 2)$
  - which are **diverse** as z is varied

y

$z \sim p_z(z)$

$x \sim \hat{p}(x \mid y)$

  - Quiz: Which color image is the ground-truth?

Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

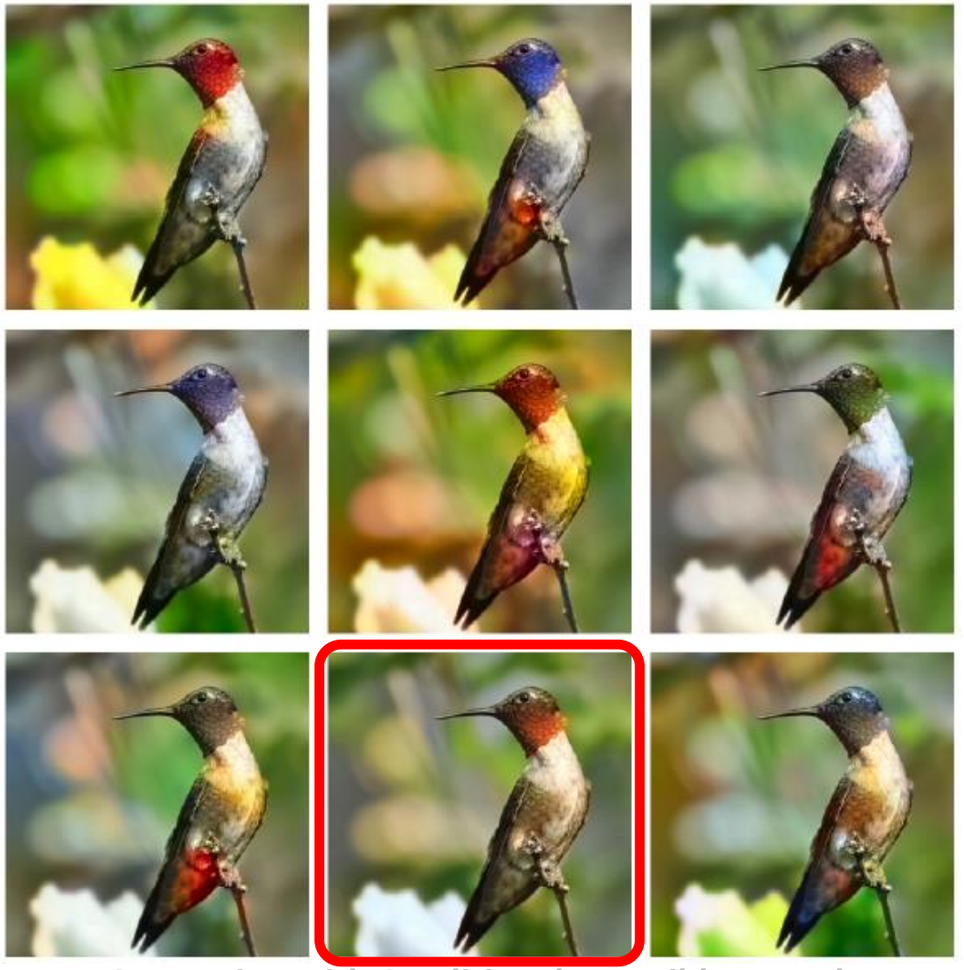
Application: Colorization

- Inverse problem:
  - given grayscale image y (256×256, L-channel in Lab color space)
  - create **realistic** color channels $\hat{x} = [a, b]$ $(64 \times 64 \times 2)$
  - which are **diverse** as z is varied

y

$z \sim p_z(z)$

$x \sim \hat{p}(x \mid y)$

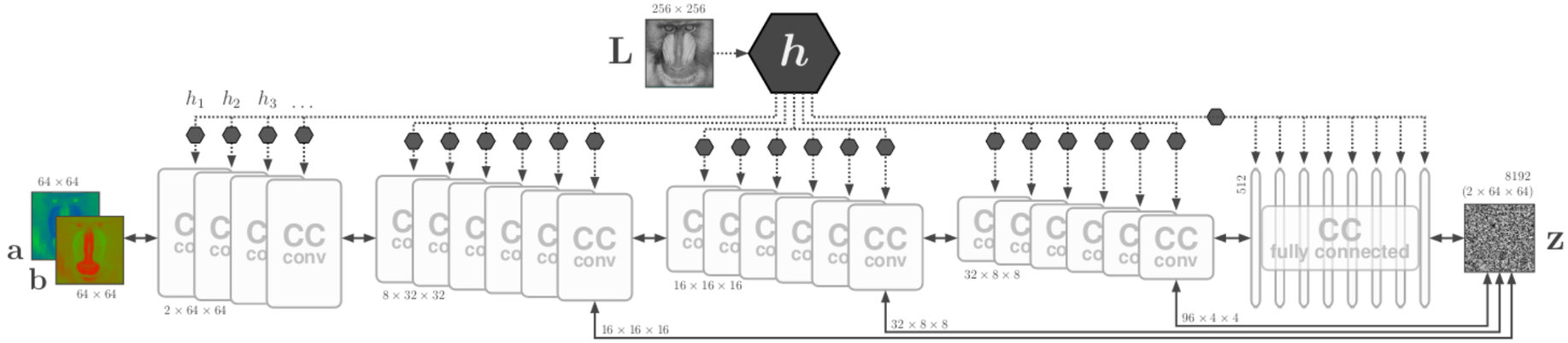


  - Quiz: Which color image is the ground-truth?

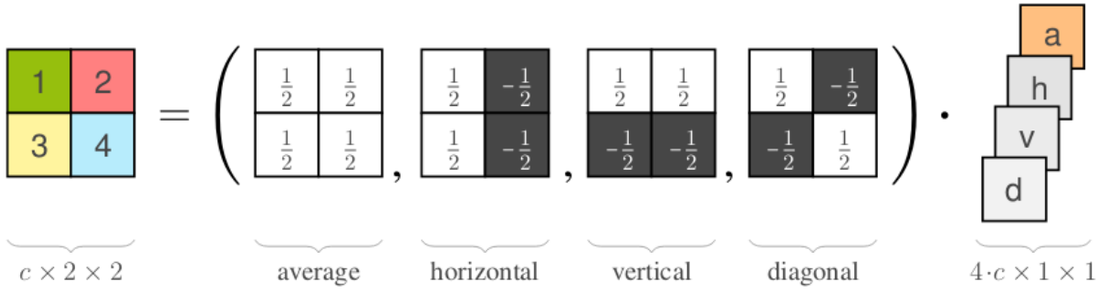
Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

cINN Architecture for Colorization

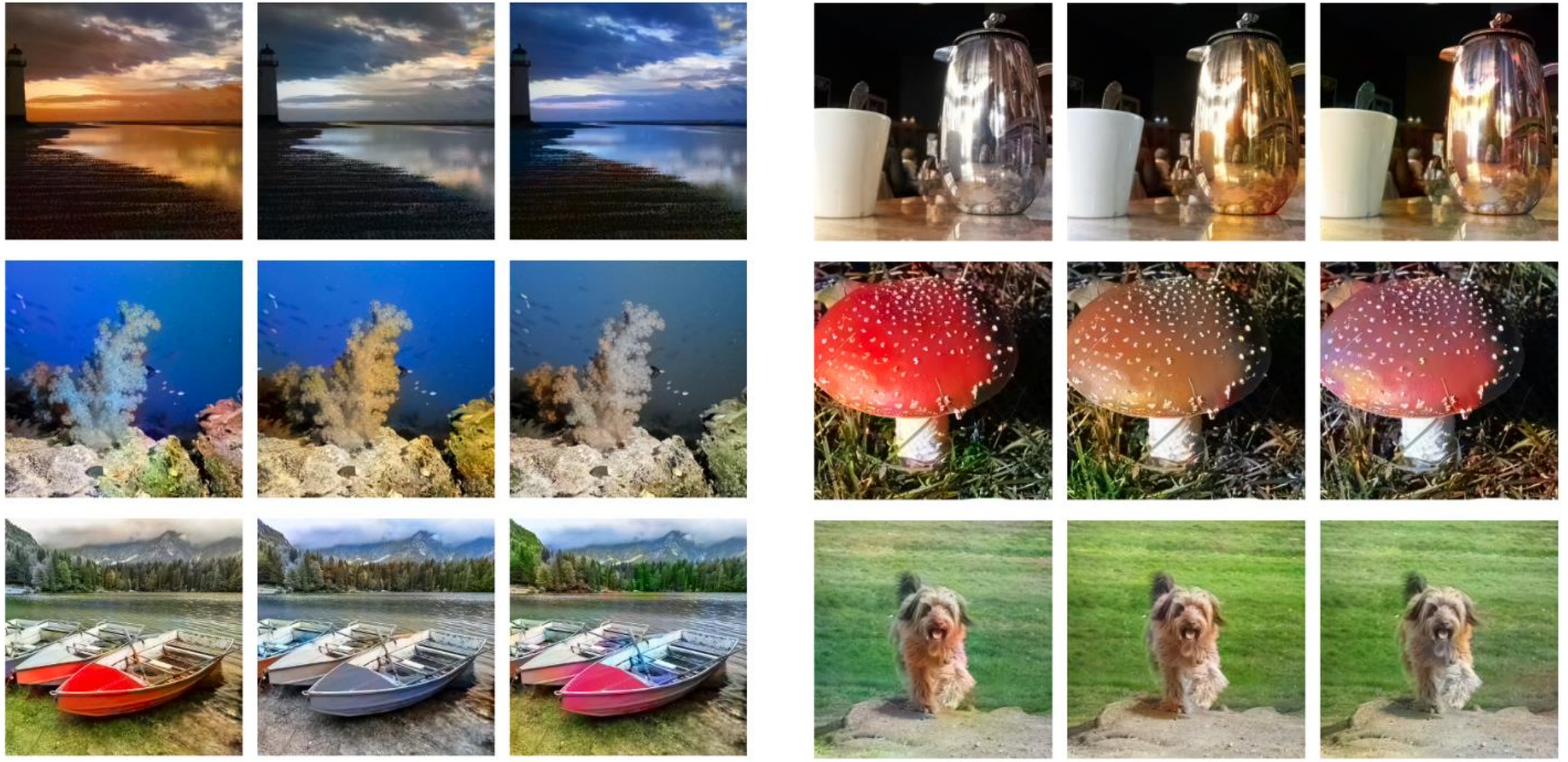
- Four convolutional stacks (with four to six coupling layers)
- Fully connected stack as backend (eight coupling layers)
- Coupling layers separated by random orthogonal matrices to mix channels
- Large feature detection network h (VGG), small conditioning networks h_l



- Multi-scale decomposition via *Haar-Wavelet* down-sampling (standard max pooling not invertible)



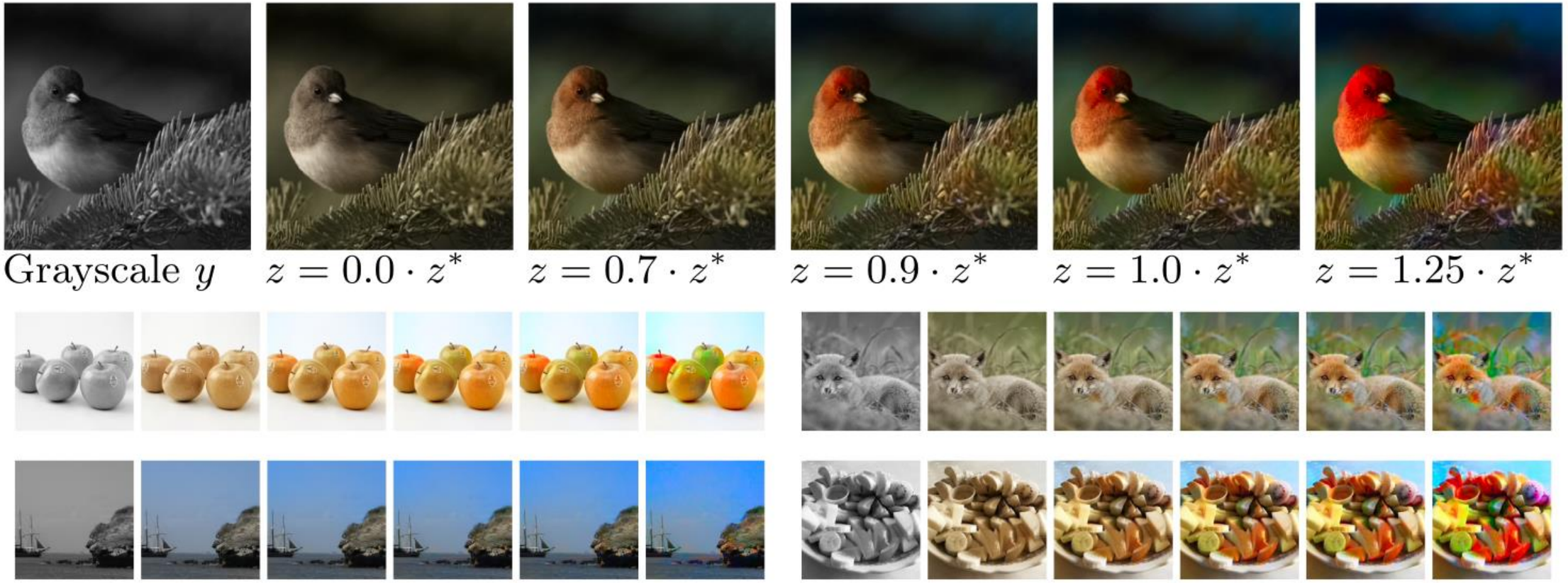
Colorization Examples



Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

Colorization: Meaningful Latent Manipulations

- Magnitude of latent vector encodes color intensity
  - Linear interpolation from $z = 0$ outwards gradually increases saturation



Colorization: Meaningful Latent Manipulations

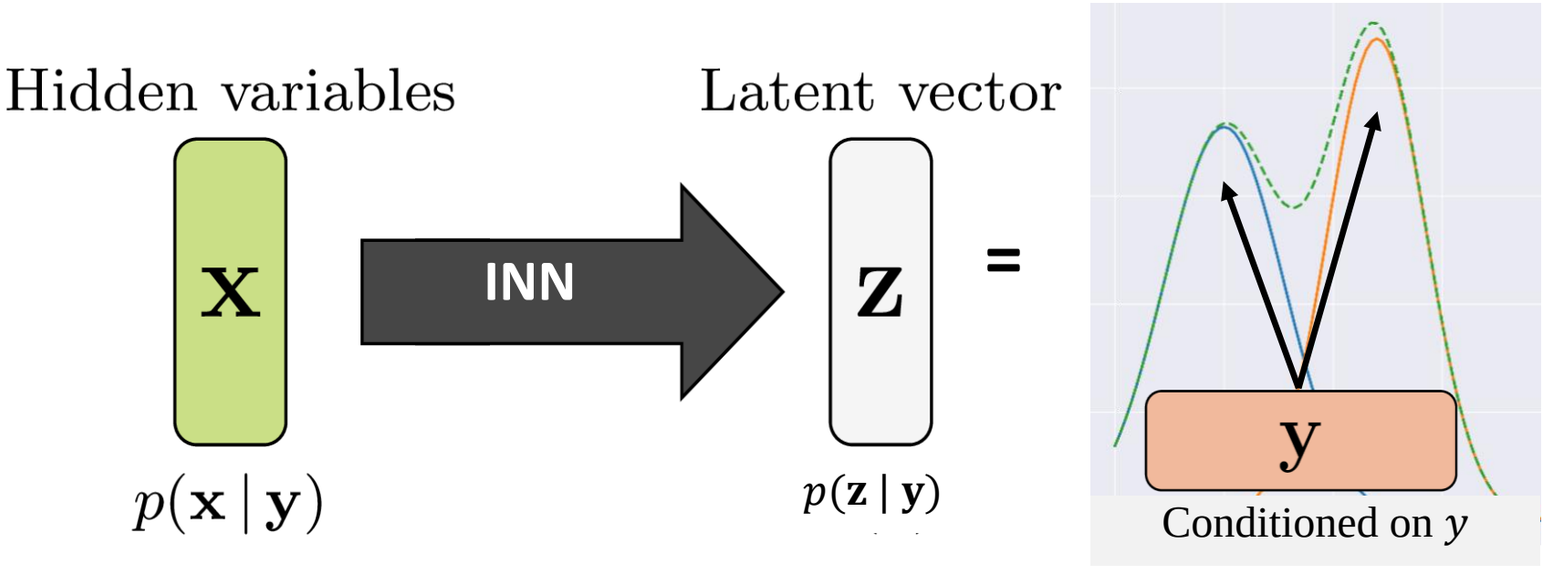
- Color transfer (in analogy to style transfer on MNIST)
  - Combining y from one image with true z from another transfers colors
    - Encode color image to latent space
    - Keep color encoding, but exchange condition (=grayscale image), then decode

Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

Application: Generative Classification

Latent Mixture INN

- Structure latent space as a mixture of Gaussians instead of a single Gaussian
  - Condition on class label *y*
    - one mixture component per class
    - for each class, learn mean and diagonal covariance matrix
    - all classes share the same INN

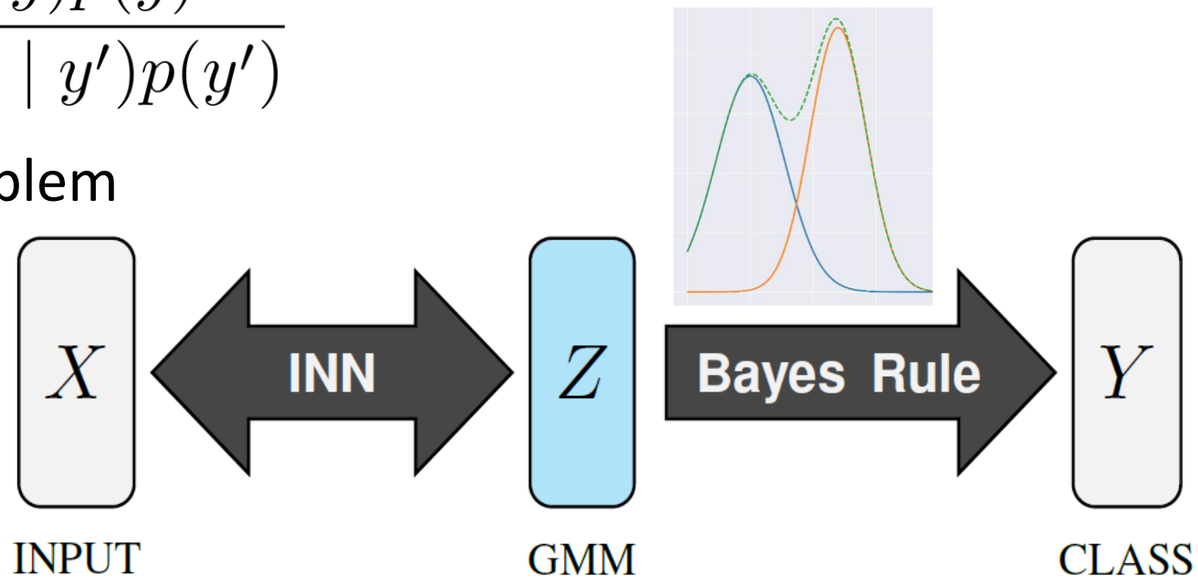


IB-INNs: Generative Classifiers

- What is a generative classifier (GC)?
  - Classifier: given image x, predict label y of most salient object
  - A discriminative classifier (DC): learns the class posterior probability $p(y \mid x)$
  - Generative classifier: instead learns the data likelihood $p(x \mid y)$ and computes the posterior indirectly by Bayes rule:

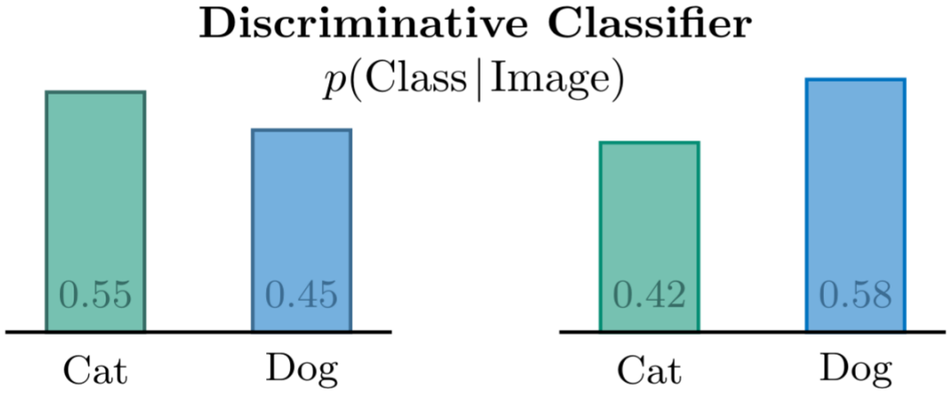
$$p(y \mid x) = \frac{p(x \mid y)p(y)}{\sum_{y'} p(x \mid y')p(y')}$$

- Learning $p(x \mid y)$ is a density estimation problem
  - Normalizing flows are good at density estimation
  - We actually model $p(z|y)$ as a latent GMM and train an INN to transform this into $p(x \mid y)$
  - The model can be trained using the Information Bottleneck Principle

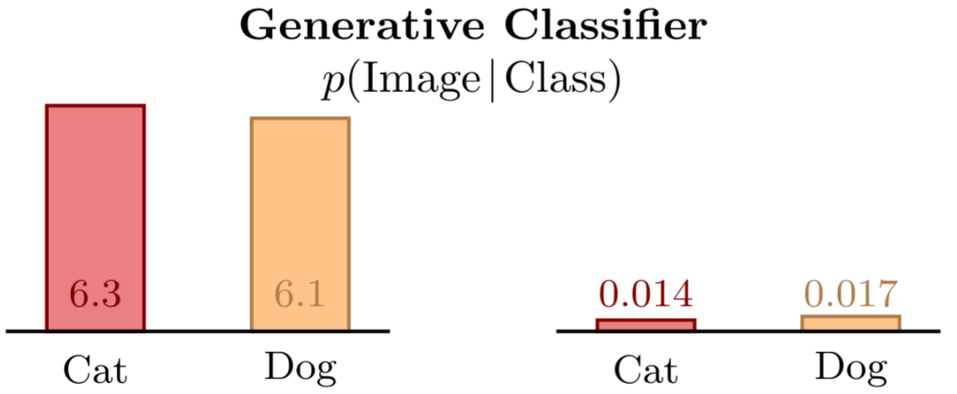


IB-INNs: Generative Classifiers

- Generative classifiers recognize out-of-distribution examples:



Always sums to 1 over classes



Does not sum to 1
(integrates to 1 over all images)

Normal input

OoD input

Normal input

OoD input

The Information Bottleneck Principle

- Naively trained generative classifiers: poor classification accuracy in comparison to DCs
  - Tend to overfit
- Information bottleneck principle overcomes this problem
  - Introduce latent representation z, where all information flows through – "bottleneck"
  - Latent variables z should be: highly informative for y (= good classification)
    keep only as much information about x as needed (= no overfitting)
- Minimize Information Bottleneck (IB) loss

$$\mathcal{L}_{\text{IB}} = I(x, z) - \beta \cdot I(y, z)$$

Generative aspect (minimize spurious information about x)

Trade-off parameter

Discriminative aspect (maximize information about class labels)

with Mutual Information (MI) $I(y, z) = D_{\text{KL}}\left(p(y, z) \middle\| p(y)p(z)\right)$

The IB Objective for INNs

- Mutual information for an GMM can be calculated analytically:

$$I(y;z) = \mathbb{E}_{p(y)}[-\log p\,(y)] + \mathbb{E}_{p(y,z)}\left[\log \frac{p(z|y)p(y)}{\sum_y p(z|y)p(y)}\right]$$

- Mutual Information $I(x;z)$ is infinite, because INN is a bijection, i.e. information preserving
  - Artificially introduce controlled information loss: $z_\epsilon = f_{\text{INN}}(x+\epsilon)$ with noise $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbb{I})$, σ small
  - Surprisingly, mutual information $I(x;z_\epsilon)$ reduces to the usual maximum likelihood loss

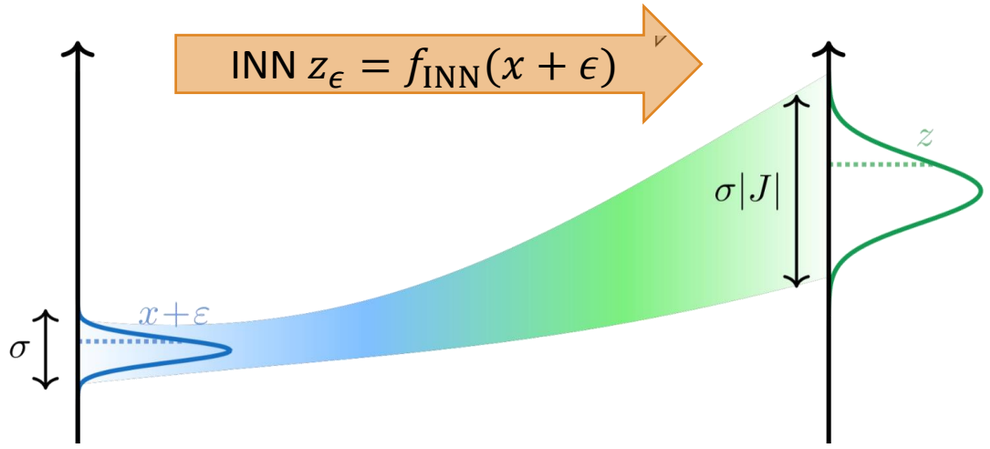
$$I(x;z_\epsilon) = \mathbb{E}_{p(x),p(\epsilon)}[-\log p\,(x+\epsilon)] = \mathbb{E}_{p(x),p(\epsilon)}[-\log p\,(z_\epsilon) - \log\,\det|J|]$$

- Intuitive effect:
  - INN amplifies the noise to minimize $I(x;z_\epsilon)$

  ⇨ x is optimally compressed in latent space (like VAE)

  ⇨ Robust generative model thanks to lack of overfitting

  ⇨ Trade-off between generative and discriminative performance by balancing $I(x;z_\epsilon)$ and $I(y;z)$



IB-INNs: Discriminative and Generative Performance

- Successfully trained on CIFAR-10 (10 classes) and ImageNet (1000 classes)

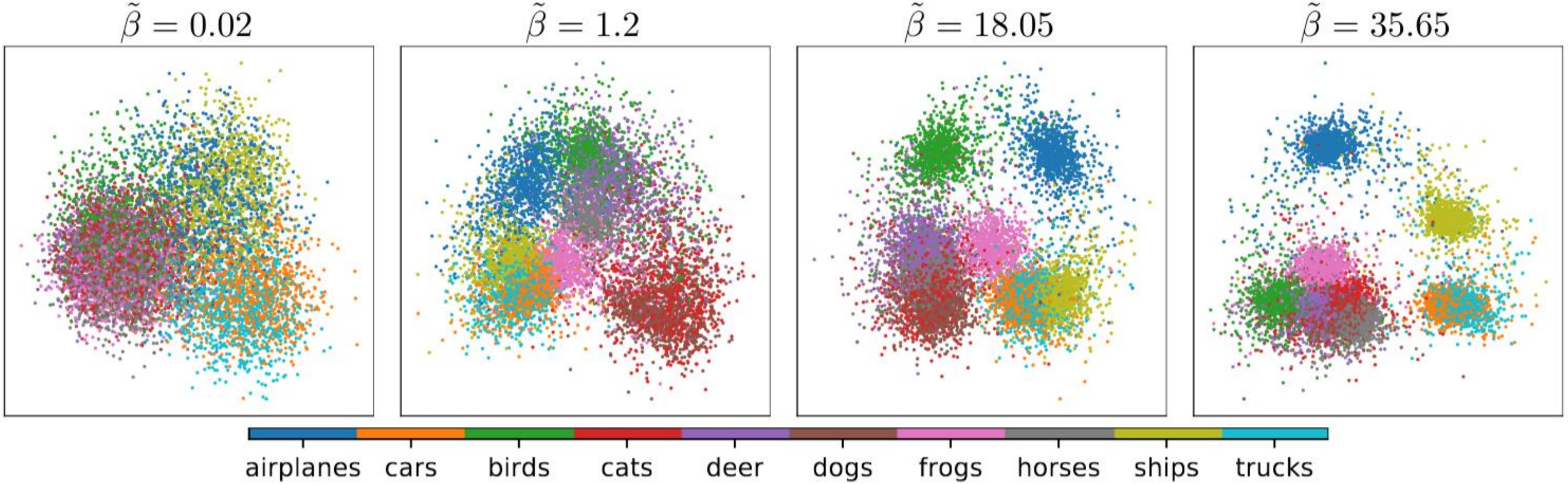
Model		Classif. err. (↓)	Bits/dim (↓)
IB-INN (ours)	$\gamma = 1$	10.27	5.25
	only $\mathcal{L}_X$ ($\gamma = 0$)	–	**4.80**
	only $\mathcal{L}_Y$ ($\gamma \to \infty$)	8.72	17.27
Stand. GC	Class-NLL	61.75	4.81
	Class-NLL + regul.	40.04	4.83
Pure DC	VIB ($\gamma = 1$)	6.83	–
	ResNet	**6.51**	–
	i-RevNet	9.22	–

β	Bits/dim. (↓)	Acc. (%) (↑)
1	4.34	67.30
2	6.14	71.73
4	8.72	73.69
8	12.35	74.59
16	17.43	75.54
32	22.68	76.18
∞	47.01	76.27
0	2.59	–
ResNet	–	77.40

Ardizzone et al. **"Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification"**, arXiv 2020.
Mackowiak, Ardizzone et al. **"Generative Classifiers as a Basis for Trustworthy Computer Vision"**, arXiv 2020.

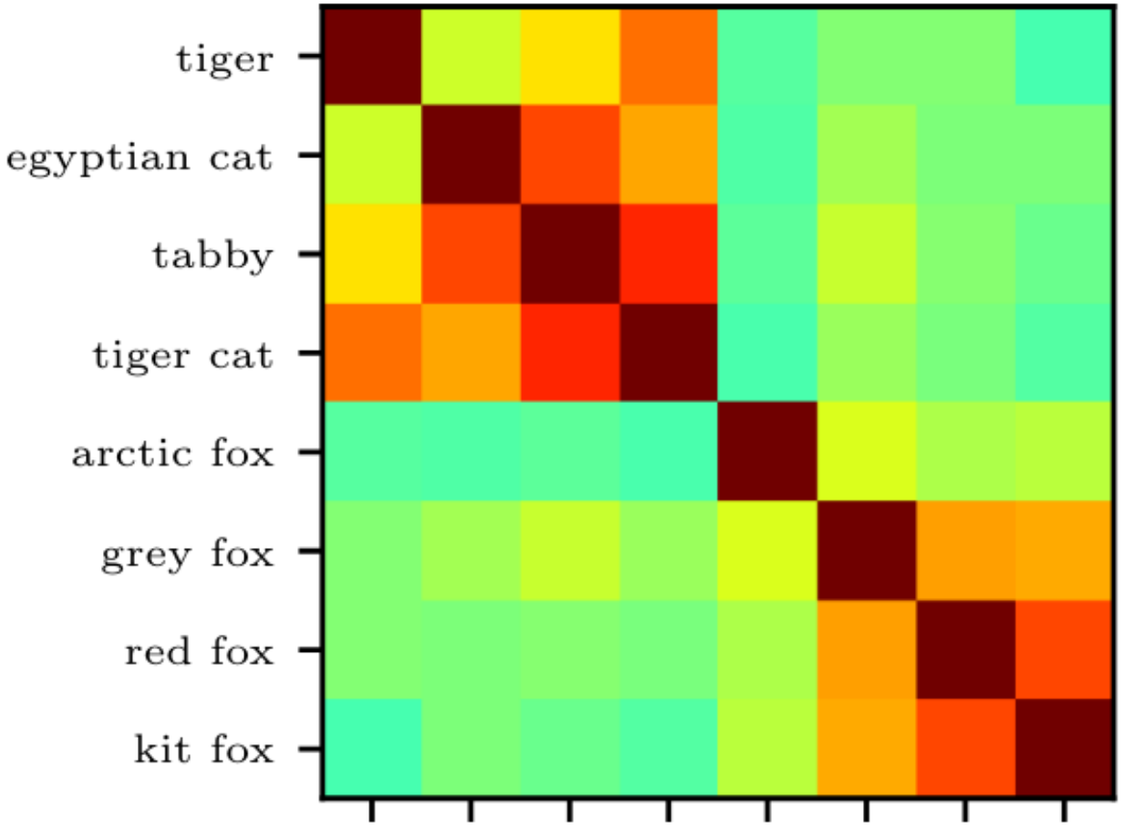
IB-INNs: Benefits of GC (1): Interpretability

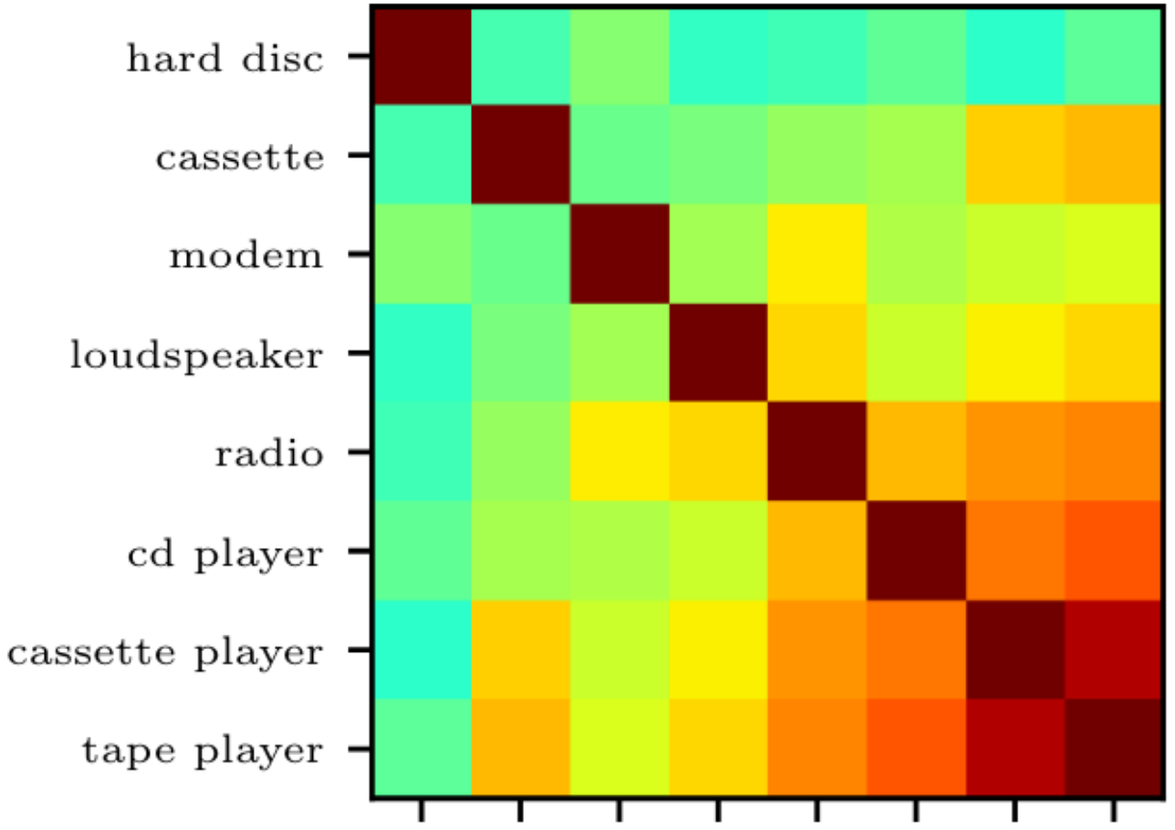
- Class separation improves as β (= importance of $I(y; z)$) increases
  - CIFAR-10 examples (PCA projection of latent space)



IB-INNs: Benefits of GC (1): Interpretability

- Pairwise distances between class centers reflect class similarity
  - ImageNet examples



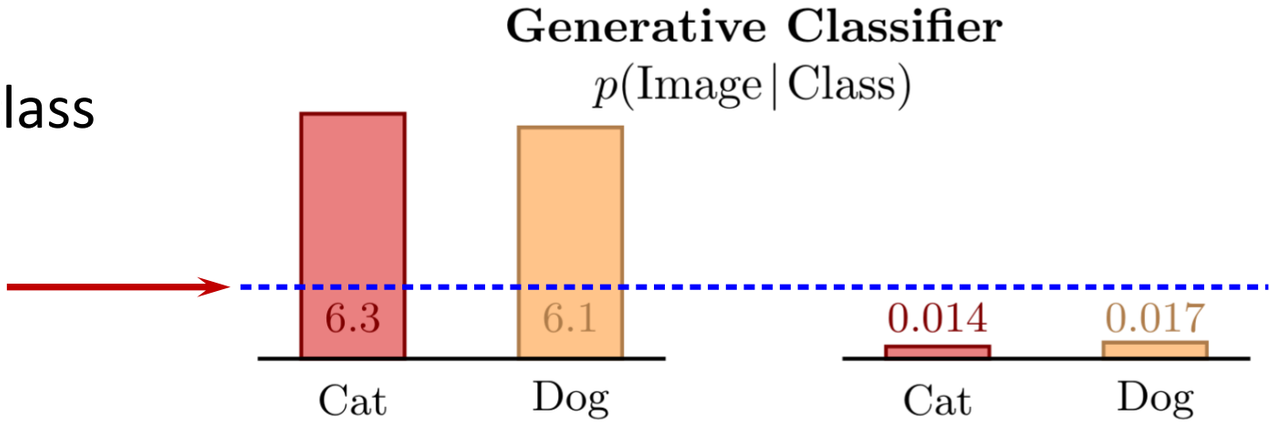


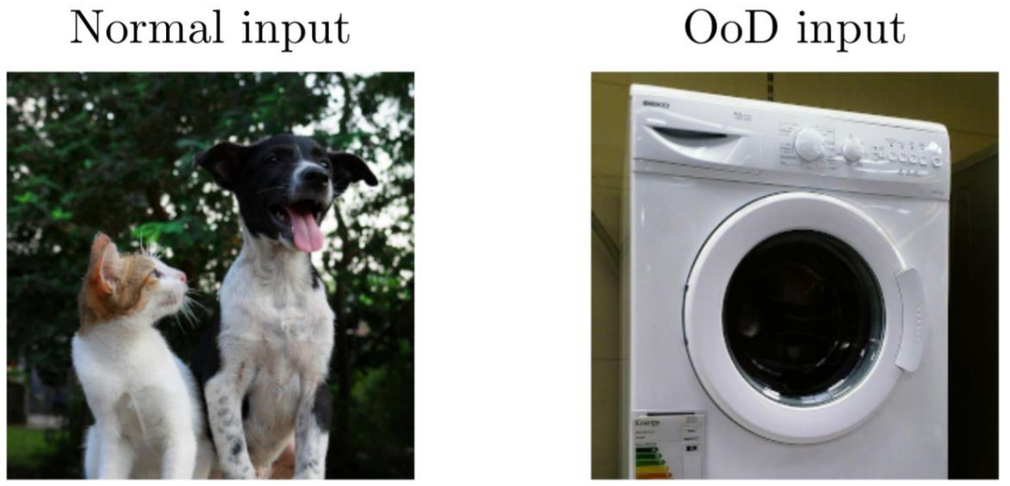
IB-INNs: Benefits of GC (1): Interpretability

- Heatmaps for attention area of most probable classes
  - ImageNet examples
  - Back-project relevant latent features of DCT pooling to image space regions
  - Due to invertibility, this represents the true decision process, not a post-hoc explanation

IB-INNs: Benefits of GC (2): Out-of-Distribution Detection

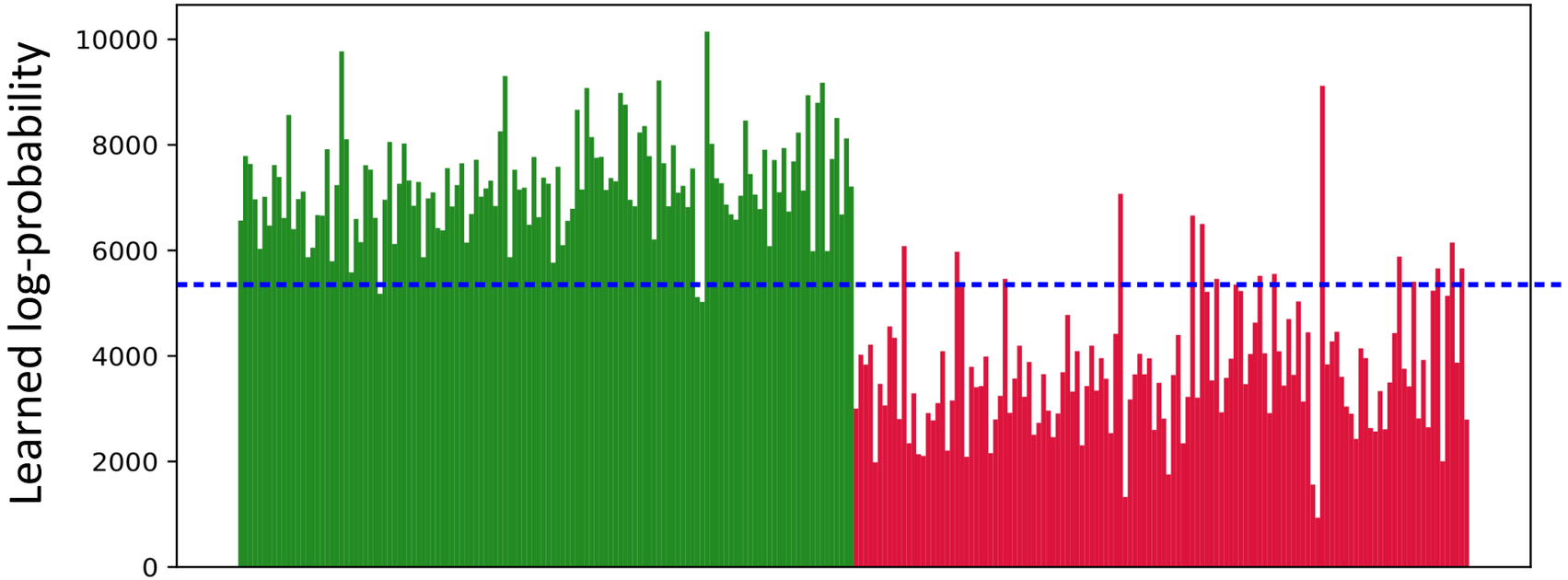
- Outliers have low likelihood for every class
  - Intuitively: can separate in-/outliers using threshold on likelihood
  - More advanced methods available
    - Typicality test
    - WAIC
    - ...
  - Many interesting open questions





IB-INNs: Benefits of GC (2): Out-of-Distribution Detection

- Outliers have low likelihood for every class
  - Artificial outliers: scrambled colors (CIFAR-10 examples)



97% outlier detection accuracy

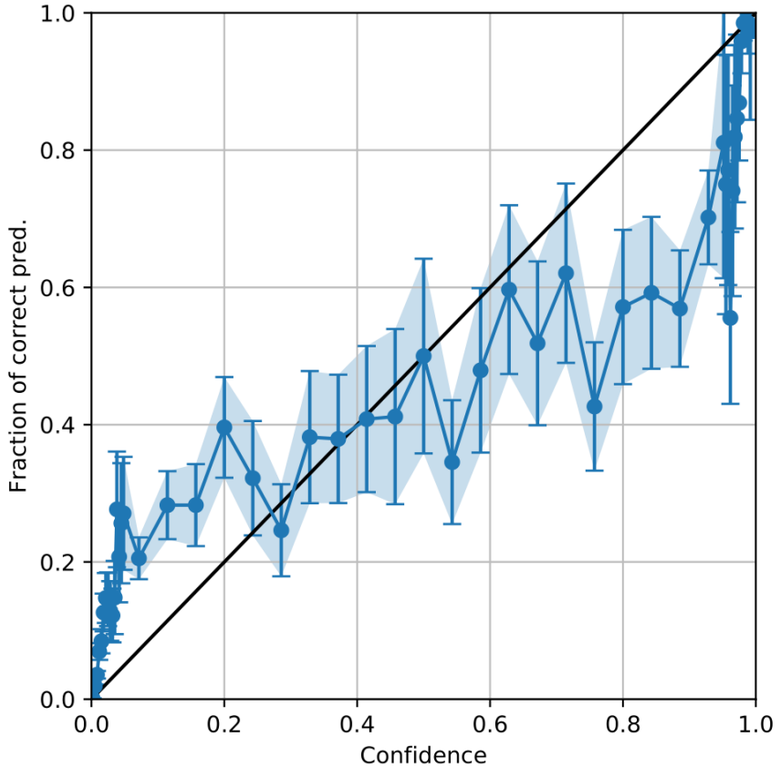
Inlier images (test set)

Same images, scrambled colors

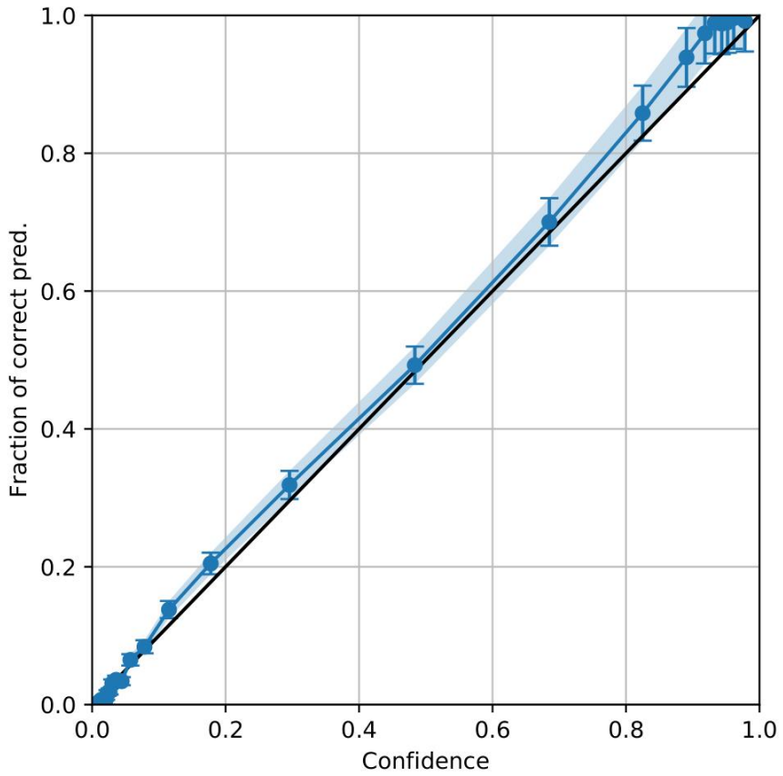
IB-INNs: Benefits of GC (3): Uncertainty Calibration

- Calibration = consistency of confidence vs. actual performance
  - If classifier is 90% confident about class label, it should be right 90% of the time, neither less nor more
  - Problematic for discriminative classifiers [Guo et al. 2017] – IB-INNs are much better calibrated

DC: ResNet-18
(CIFAR-10)



GC: IB-INN β=1
(CIFAR-10)



Guo et al. **"On calibration of modern neural networks"**, ICML 2017.
Ardizzone et al. **"Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification"**, arXiv 2020.

Guaranteed disentanglement with Nonlinear ICA and Incompressible Flows

What is Disentanglement?

- Latent dimensions should have one and only one isolated effect on the data

β-VAE

VAE

- β-VAE disentangles azimuth whereas VAE entangles it with other variables

Higgins et al. **"beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework",** ICLR 2017.

ID-GAN

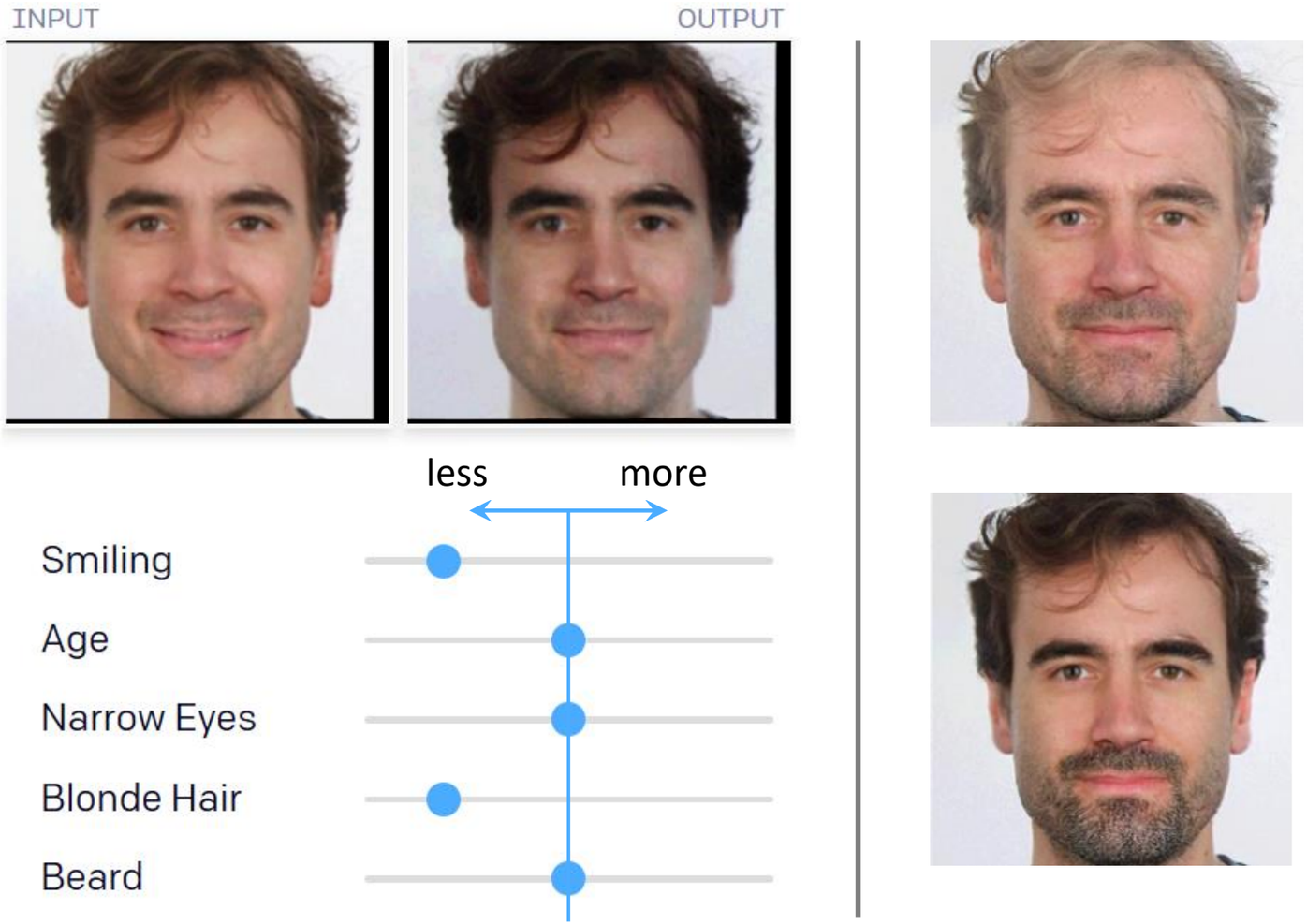
- Information-Distillation Generative Adversarial Network is probably state-of-the-art
- Combine VAE encoder with conditional GAN generator
  - Works well on large images (CelebA-HQ: 1024x1024)
  - GAN conditioned on β-VAE latent code
  - Additional cycle constraint: maximize mutual information between latent codes of real and fake images

Lee et al. **"High-Fidelity Synthesis with Disentangled Representation",** arXiv 2020.

GLOW (again)

- Learns disentangled representations using a normalizing flow

- Try your own face: openai.com/blog/glow/



Disentanglement: Definition

- Definition by Bengio et al.:
  - A disentangled representation has recovered the *"informative factors of variation"* in a dataset
  - Disentangled latent features separate different categories of information (e.g. identity, pose and background) into independent degrees of freedom

- Disentangled representations are interpretable by humans and generalize well for downstream tasks and transfer learning

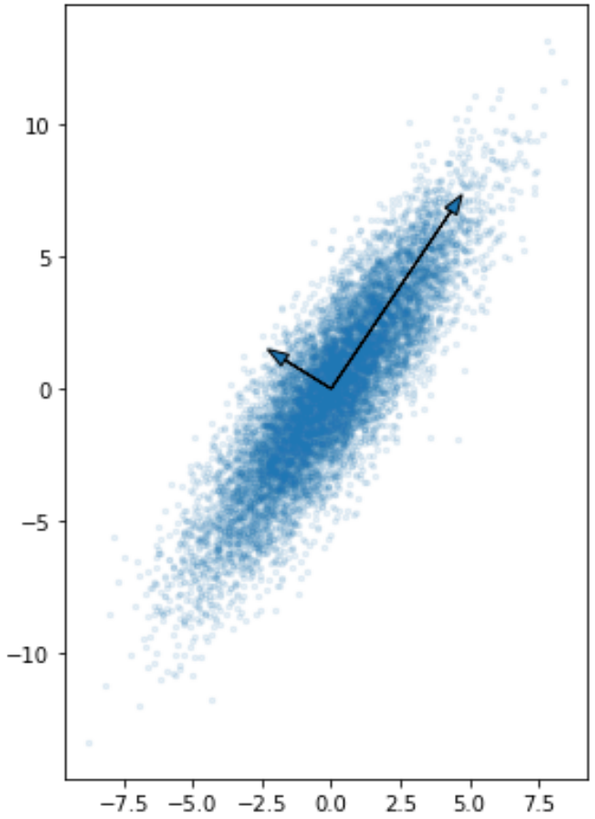
- Methods so far empirically work well, but have no theoretical guarantees

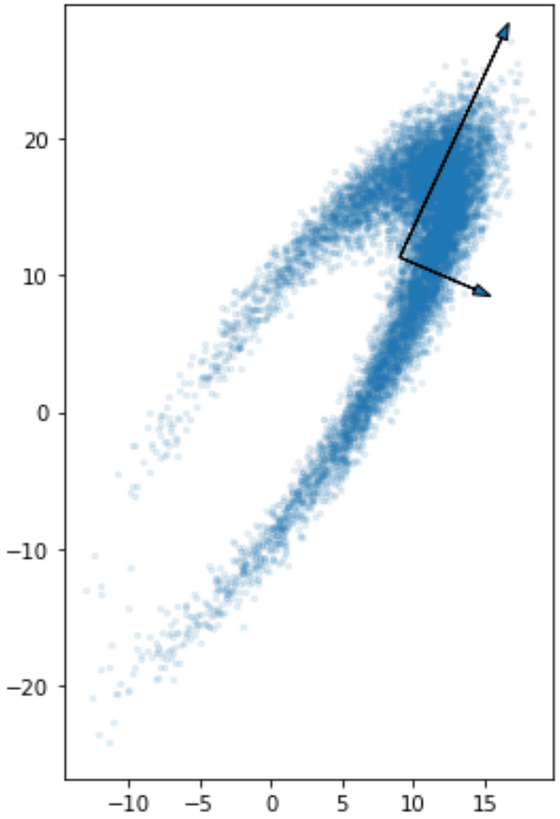
- We apply the theory of nonlinear ICA to INNs to derive such guarantees

Bengio et al. **"Representation Learning: A Review and New Perspectives",** PAMI 2013.
Khemakhem et al. **"Variational autoencoders and nonlinear ICA: A unifying framework",** AISTAT 2020.

Recap: PCA (Principal Component Analysis)

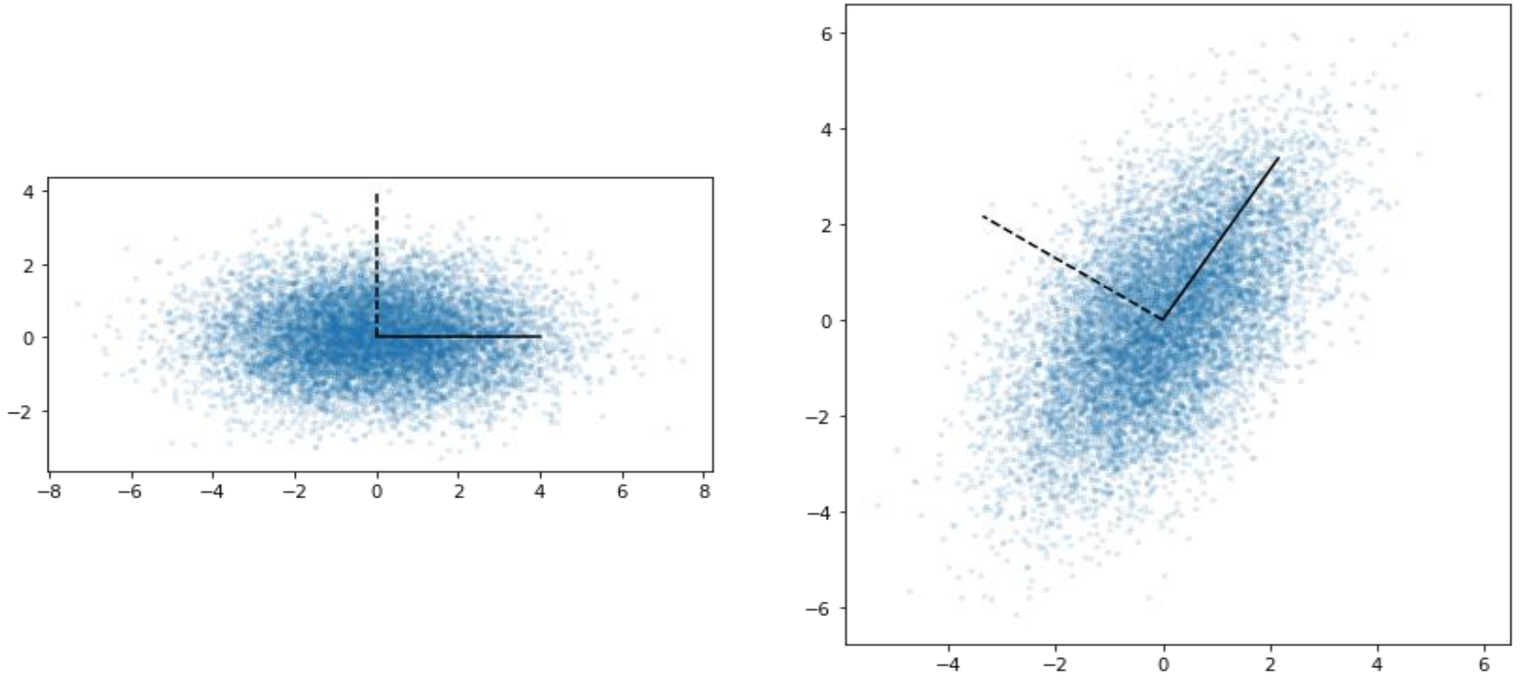
- Classical method for unsupervised disentanglement with a linear transformation
  - Finds uncorrelated basis vectors for multivariate Gaussian distributions
  - Can be applied to non-Gaussian data, but cannot fully disentangle them





PCA as a Generative Model

- Rotate uncorrelated Gaussian in latent space into desired distribution in data space
  - Special property of Gaussians: uncorrelated = statistically independent / factorial



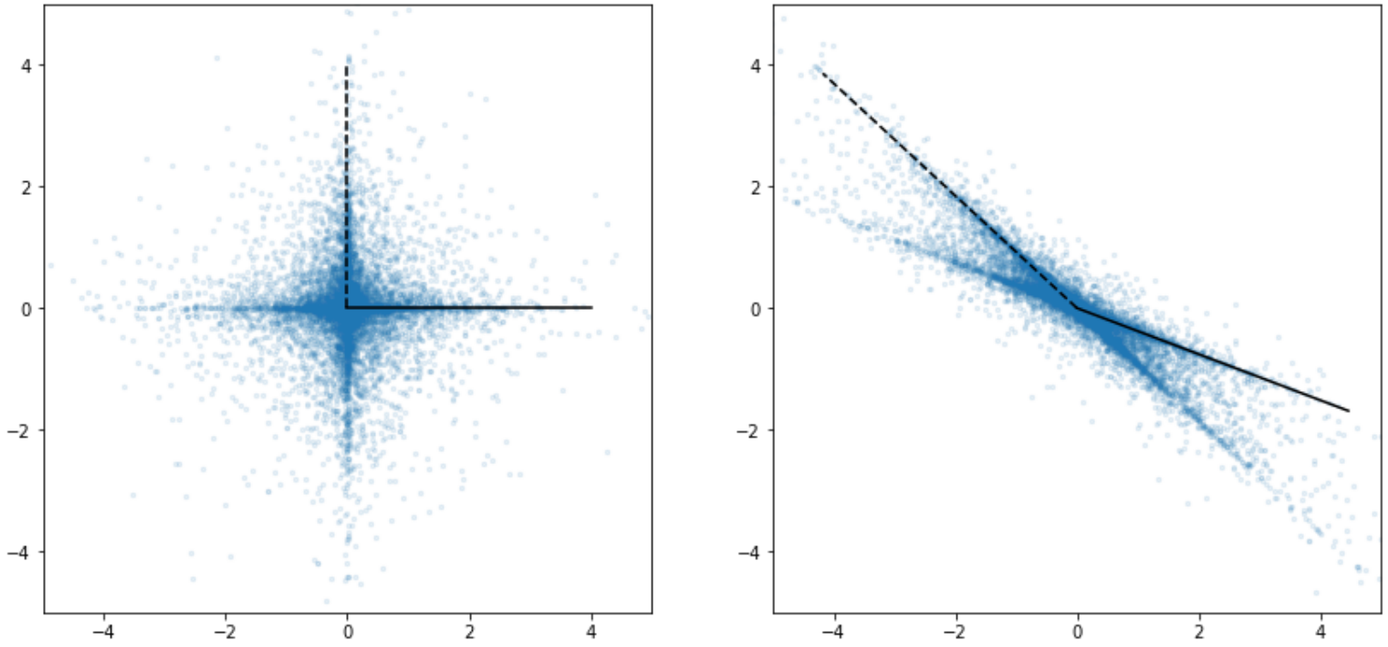
Latent space
(uncorrelated gaussian)

Rotation

Data space
(correlated gaussian)

ICA as a Generative Model

- Roughly: Independent Component Analysis generalizes PCA to non-gaussian case
  - Apply arbitrary invertible linear transformation to factorial non-Gaussian latent distribution



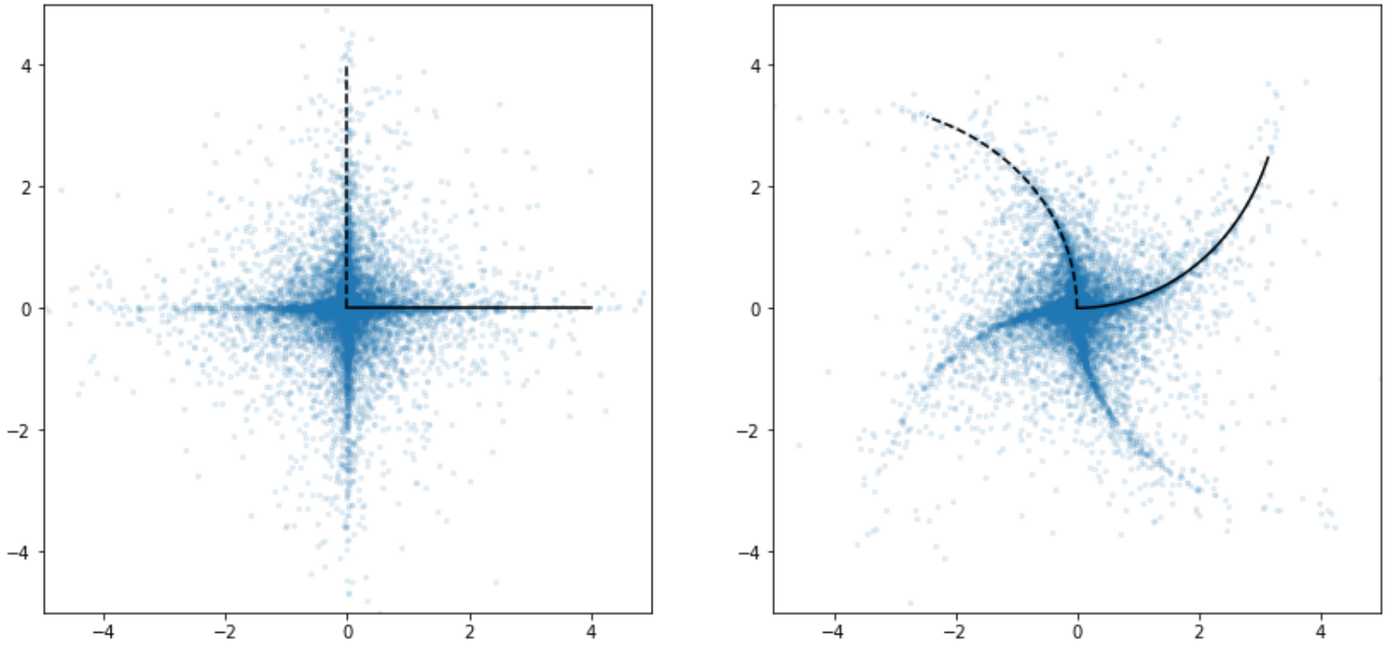
Latent space
(non-gaussian,
independent dimensions)

Invertible linear
transformation

Data space

Nonlinear ICA as a Generative Model

- Replace the linear transformation with an invertible non-linear transformation
- Hyvärinen proved: generally unidentifiable (infinitely many possible solutions)



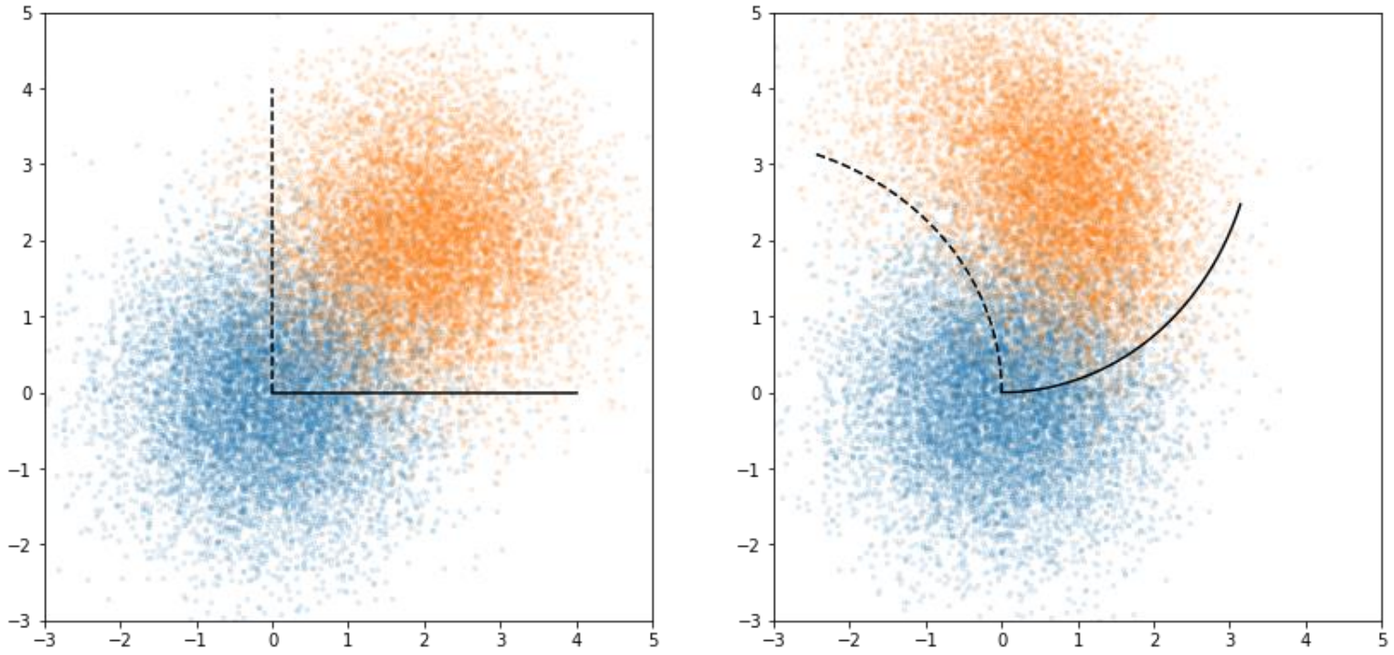
Latent space

Invertible non-linear transformation

Data space

ICA as a Disentanglement Method

- Fundamental insight: we need to constrain transformations *g* in the latent space
  - Constrain latent distributions by conditioning, e.g. by introducing a mixture distribution



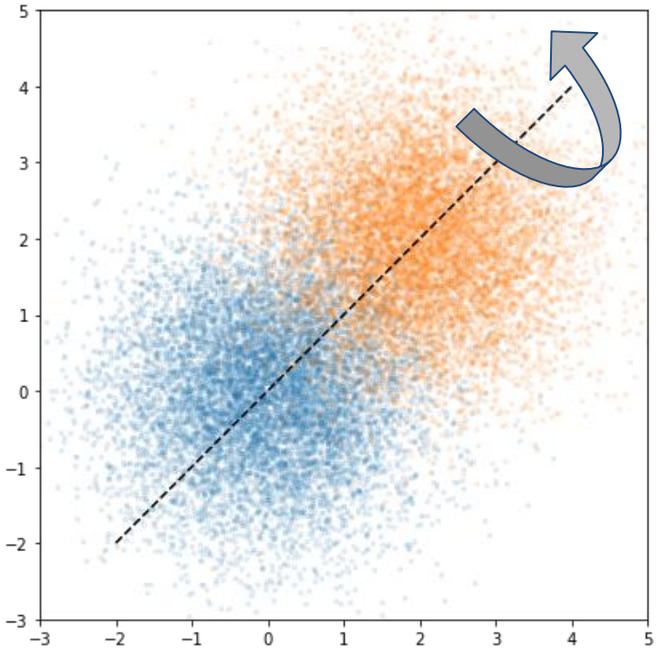
Invertible nonlinear transformation

Original latent space (conditionally independent dimensions)

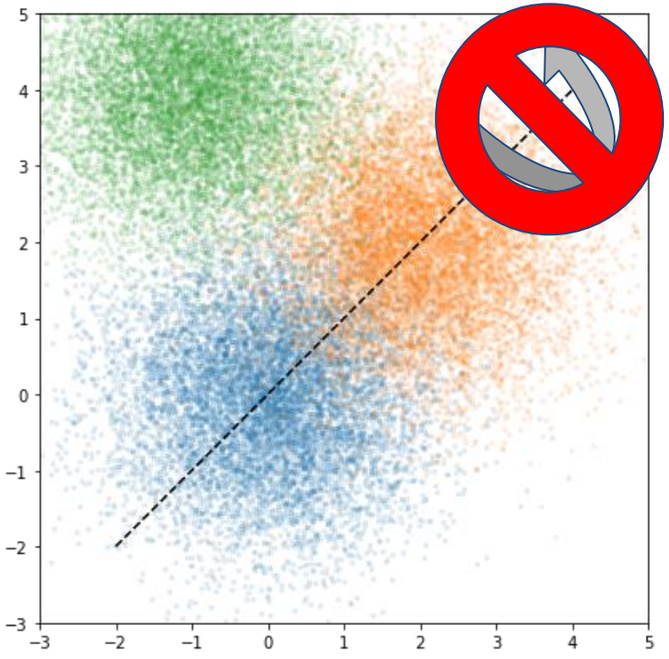
Alternative latent space (orange distribution does *not* have independent dimensions)

ICA as a Disentanglement Method

- Fundamental insight: we need to constrain transformations *g* in the latent space
  - Each additional condition constrains a degree of freedom



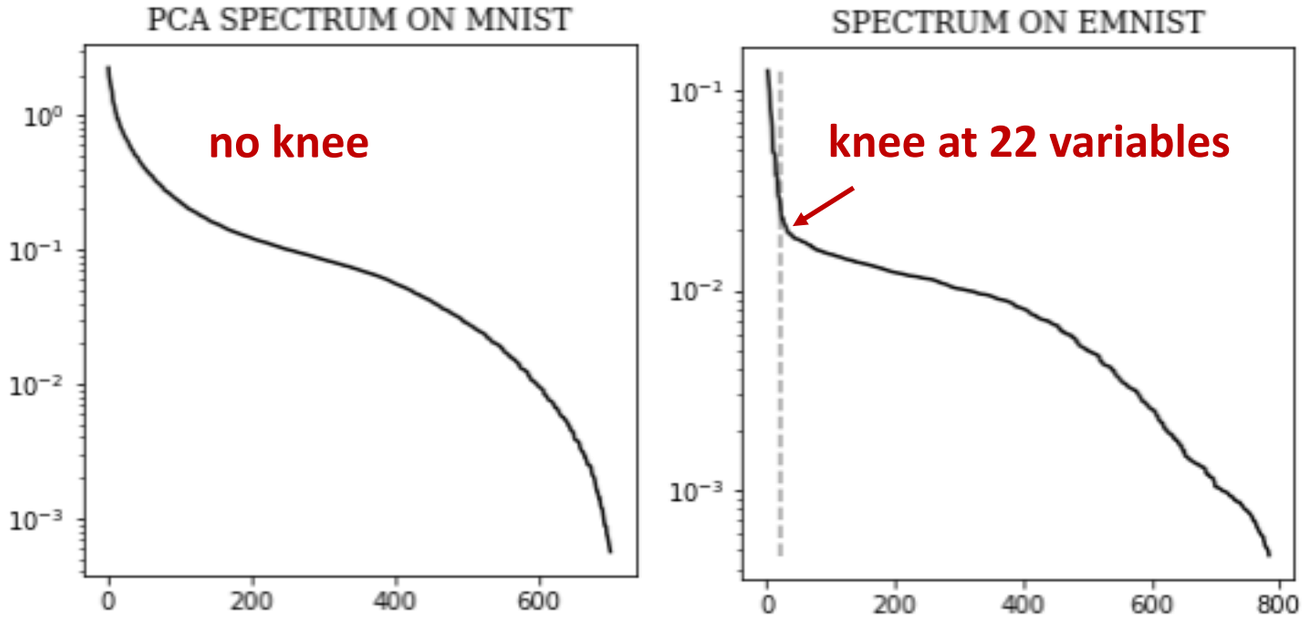
We can rotate around the axis through the means and still have conditionally independent distributions

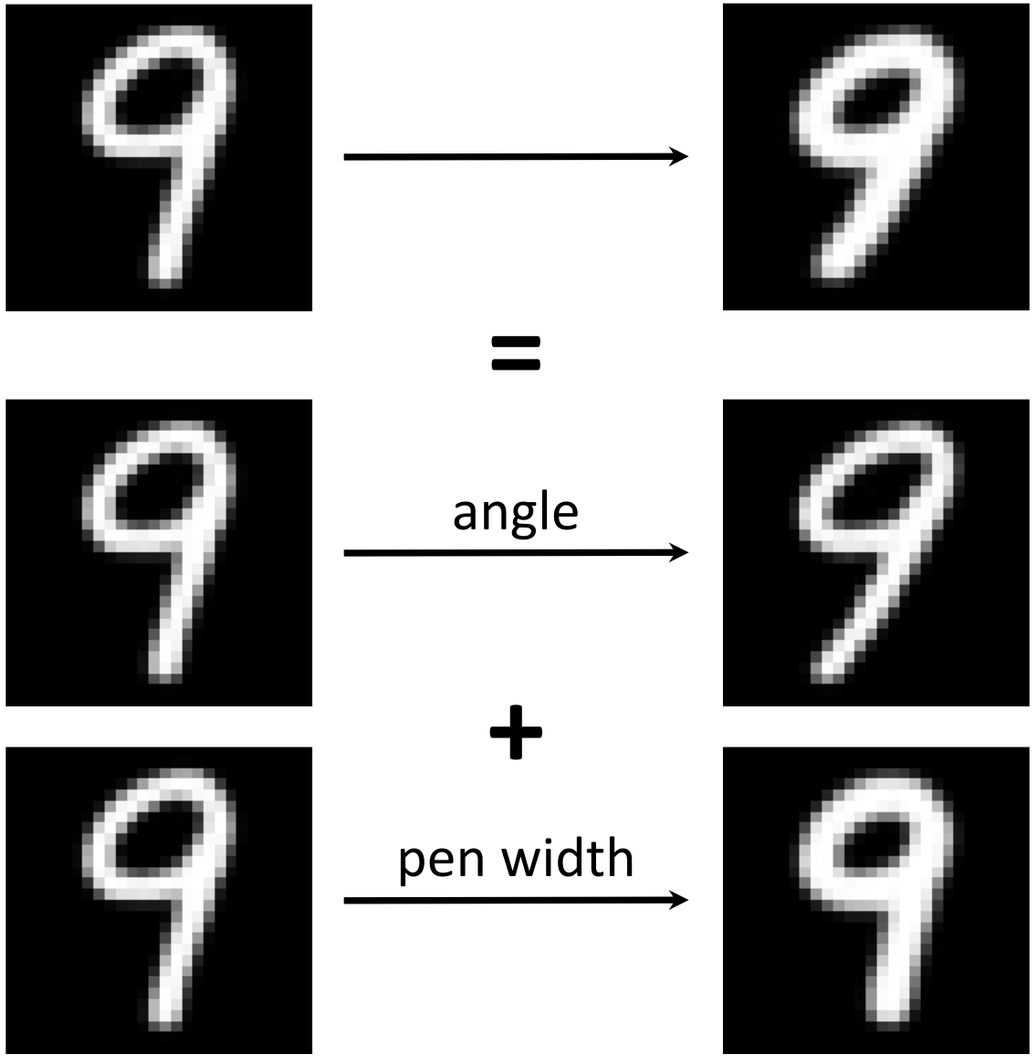


After adding a third component, rotation no longer preserves the distribution (when means are in general position)

Application to EMNIST

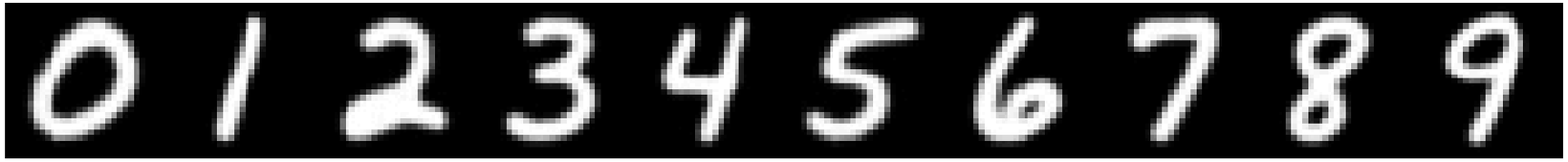
- Express complex changes of handwritten digits as a combination of simple changes
- Identify intrinsic variables, which compactly and intuitively explain variability



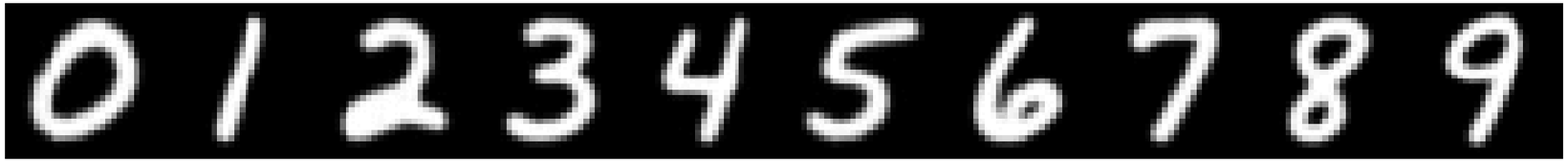


Application to EMNIST

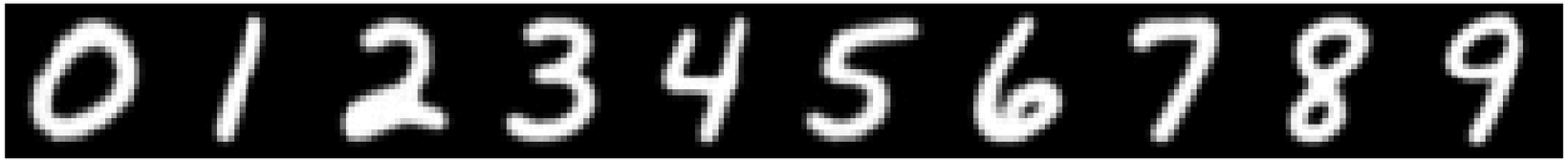
- First 8 latent variables control global properties



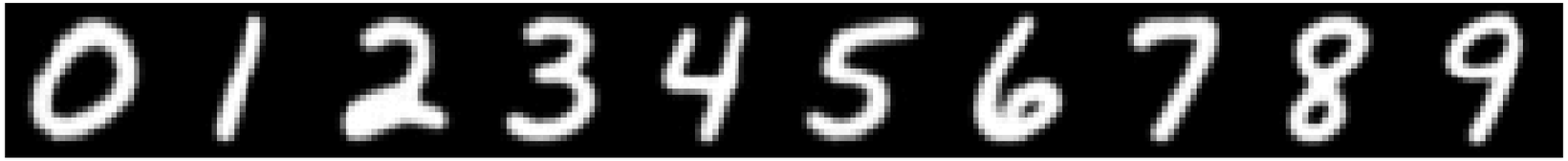
Slant



Height



Width of top



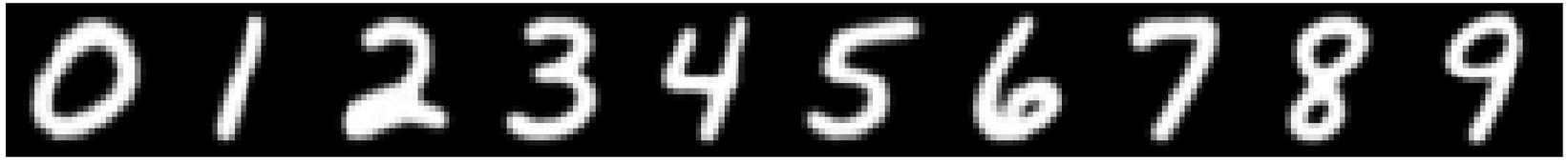
Width of bottom

(animations not visible in PDF version)

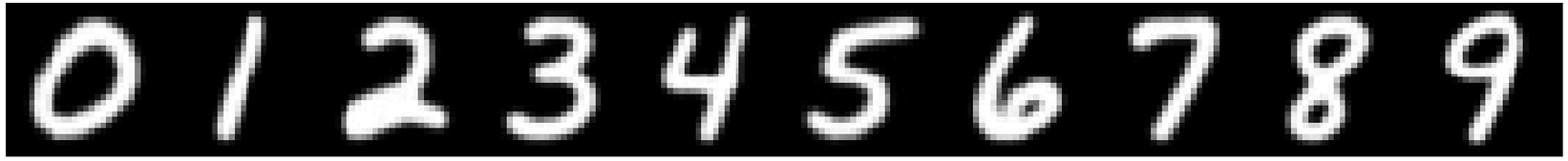
Application to EMNIST

- First 8 latent variables control global properties

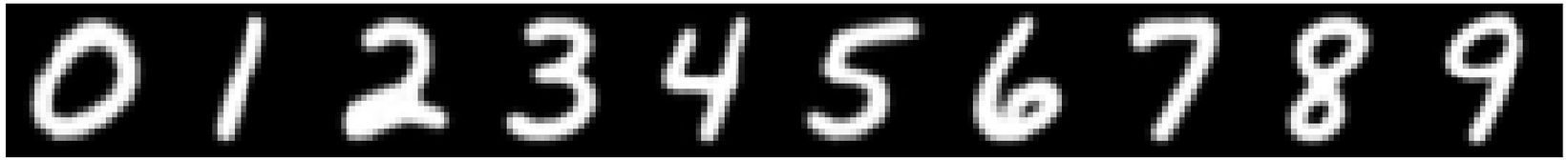
- Following 14 control local shape



Openness of lower loop



Extension to top right

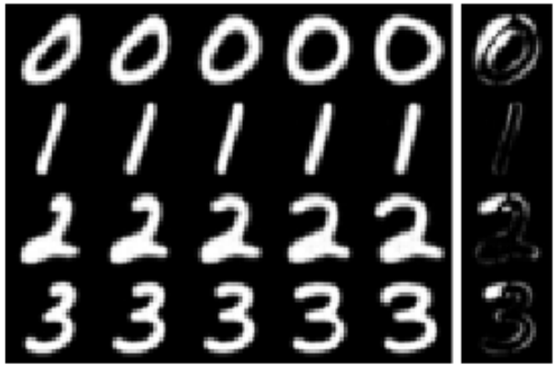


Tail of 2

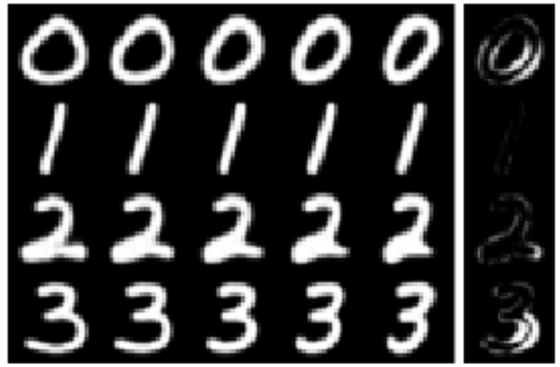
(animations not visible in PDF version)

Application to EMNIST

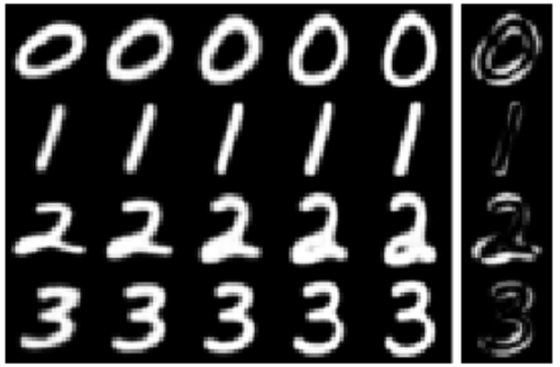
- First 8 latent variables control global properties
- Following 14 control local shape
- Remaining 762 have no visible effect



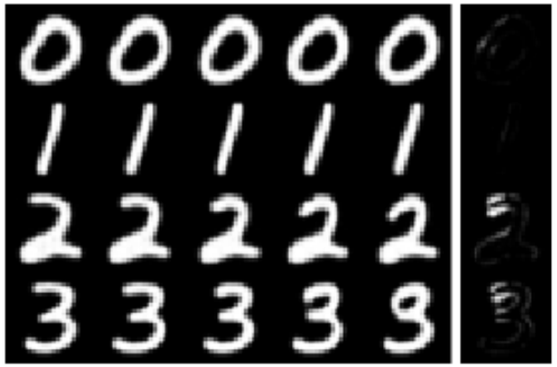
Var. 1: upper width



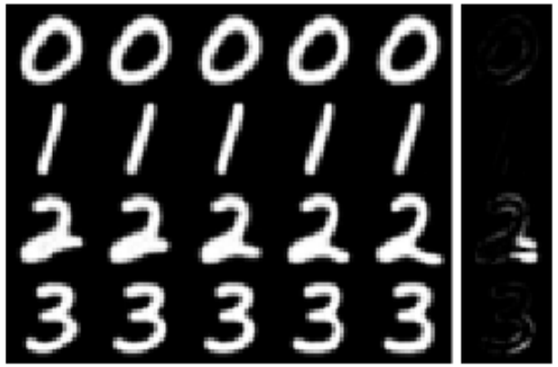
Var. 8: lower width



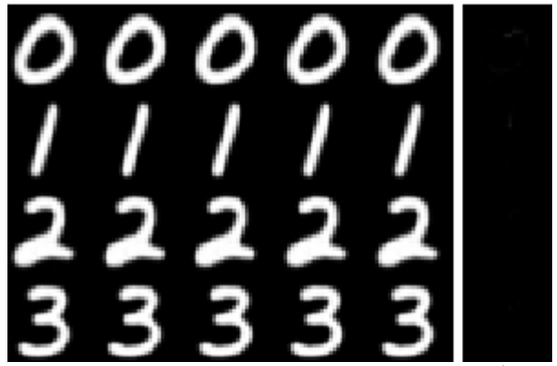
Var. 3: height



Var. 13: top left of 2,3,7



Var. 16: tail of 2



Var. 23: no effect

difference images

Application to EMNIST

Latent space interpolation

Independent effect of first 8 most significant latent dimensions

(animations not visible in PDF version)

Summary

Public code of our FrEIA library: https://github.com/VLL-HD/FrEIA

- INNs are very good density estimators:
  - Not yet quite as good as GANs (as trained by the Big Guys with 300 GPUs in parallel ☺)
  - But with much stronger mathematical interpretation and guarantees
  - Successful applications in particle and astro-physics, epidemiology, medicine, psychology
- Three main approaches to incorporate additional information
  - Conditional INN: learn $p_z(\mathbf{z} = f_{\text{INN}}(\mathbf{x}; \mathbf{y}))$
  - Latent mixture INN: learn $p_z(\mathbf{z} = f_{\text{INN}}(\mathbf{x}) \mid \mathbf{y})$
  - Augmented latent space INN: learn $p_{y,z}(\mathbf{y}, \mathbf{z} = f_{\text{INN}}(\mathbf{x}))$
  - We get the full posterior $p(x \mid y)$, both exactly and through samples
- Future work:
  - Improve architectures and training
  - Strengthen validation and mathematical guarantees
  - More applications in natural, social, and life sciences
  - Better incorporation of prior knowledge from the application domain

Thanks to our team and collaborators!

Visual Learning Lab, Uni Heidelberg:

Lynton Ardizzone
Jakob Kruse
Jens Müller
Felix Draxler
Radek Mackowiak
Peter Sorrenson
Carsten Rother

York University, Canada:

Marcus Brubaker

German Cancer Research Center, Heidelberg:

Tim Adler, Sebastian Wirkert, Lena Maier-Hein

Depart. of Physics and Astronomy, Uni Heidelberg:

Victor Ksoll, Ralf Klessen
Anja Butter, Armand Rousselot, Tilman Plehn

Inst. for Environmental Physics, Uni Heidelberg:

André Butz, Florian Kleinicke

Psychologisches Institut, Uni Heidelberg:

Stefan Radev, Ulf Mertens, Andreas Voss

References

Adler, T., et al.: “Uncertainty-aware performance assessment of optical imaging modalities with invertible neural networks”, Intl. J. Computer Assisted Radiology and Surgery 14(6):997–1007 (2019).

Ardizzone, L., et al.: “Analyzing inverse problems with invertible neural networks”, ICLR 2019, arXiv:1808.04730 (2018).

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., & Köthe, U.: “Guided Image Generation with Conditional Invertible Neural Networks”, arXiv:1907.02392 (2019).

Ardizzone, L., Mackowiak, R., Kruse, J., Rother, C., & Köthe, U.: “Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification”, arXiv:2001.06448 (2020).

Behrmann, J., Duvenaud, D., & Jacobsen, J. H.: “Invertible residual networks”, ICML 2019, arXiv:1811.00995 (2018).

Bellagente, M. et al.: “Invertible Networks or Partons to Detector and Back Again”, arXiv:2006.06685 (2020).

Bengio, Y., et al.: “Representation Learning: A Review and New Perspectives”, IEEE PAMI 35(8):1798-1828 (2013).

Bloem-Reddy, B., & The, Y.W.: “Probabilistic symmetry and invariant neural networks”, JMLR 21(90):1−61(2020).

Chen, R., et al.: “Residual Flows for Invertible Generative Modelling”, NeurIPS (2019).

Dinh, L., Sohl-Dickstein, J., Bengio, S.: “Density estimation using Real NVP”, arXiv:1605.08803 (2016).

Gresele, L., et al.: “Relative gradient optimization of the Jacobian term”, ICML Workshop INNF+, arXiv:2006.15090 (2020).

Gomez, A., et al.: “The Reversible Residual Network: Backpropagation Without Storing Activations”, NIPS (2017).

Gouk, H., et al.: “Regularisation of Neural Networks by Enforcing Lipschitz Continuity”, arXiv:1804.04368 (2018).

Gretton, A., et al.: "A kernel two-sample test.“, JMLR 13:723-773, (2012).

Guo, Ch., et al.: “On calibration of modern neural networks”, ICML (2017).

References

He, K., et al.: “Deep residual learning for image recognition”, CVPR (2016).

Higgins, I., et al.: “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”, ICLR (2017).

Khemakhem, I., et al.: “Variational autoencoders and nonlinear ICA: A unifying framework”, AISTAT (2020).

Kingma, D., & Dhariwal, P.: “Glow: Generative flow with invertible 1x1 convolutions”, NIPS (2018).

Kruse, J., et al.: “Benchmarking Invertible Architectures on Inverse Problems”, ICML Workshop INNF (2019).

Ksoll, V. et al.: “Stellar Parameter Determination from Photometry using Invertible Neural Networks”, to appear in: MNRAS, arXiv:2007.08391 (2020).

Lee, W., et al.: “High-Fidelity Synthesis with Disentangled Representation”, arXiv:2001.04296 (2020).

Mackowiak, R., Ardizzone, L. et al.: “Generative Classifiers as a Basis for Trustworthy Computer Vision”, arXiv:2007.15036 (2020).

Putzky, P., & Welling, M.: “Invert to Learn to Invert”, NeurIPS (2019).

Radev, S., et al.: “BayesFlow: Learning Complex Stochastic Models with Invertible Neural Networks”, arXiv:2003.06281 (2020).

Radev, S., et al.: “Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany”, to appear (2020).

Shiono, T.: “Estimation of Agent-Based Models Using Bayesian Deep Learning Approach of BayesFlow”, SSRN:3640351 (2020).

Sorrenson, P., Rother, C., Köthe, U.: “Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)”, ICLR 2020, arXiv:2001.04872 (2020).

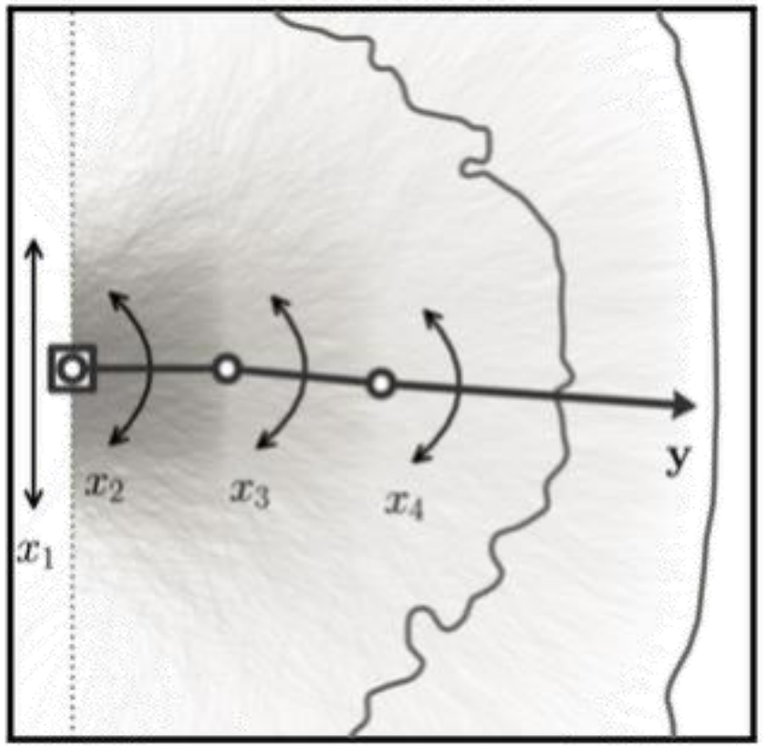
Tishby, N., et al.: “The Information Bottleneck Method”, Allerton Conf. Communications, Control, Computing, arXiv:0004057 (1999).

Non-linear Toy Example: Inverse Kinematics

Forward problem

- robot arm with 4 DOF: $x = [x_1, \dots, x_4]$
  - x_1: vertical position of first joint
  - x_2, x_3, x_4: joint angles
  - Gaussian priors: x_1 prefers the center
    x_2, x_3, x_4 prefer to be straight
- observation: hand position $y = [y_1, y_2]$
- geometric arm simulation $y = g(x)$
  implicitly defines likelihood $p(y \mid x)$

prior distribution $p(x)$



Non-linear Toy Example: Inverse Kinematics

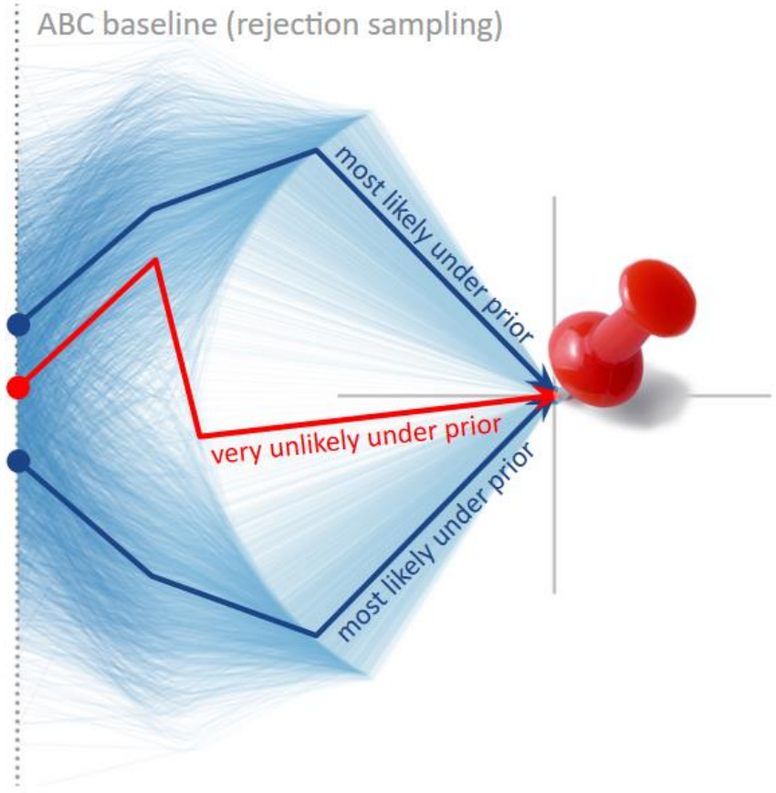
Forward problem

- robot arm with 4 DOF: $x = [x_1, \dots, x_4]$
  - x_1: vertical position of first joint
  - x_2, x_3, x_4: joint angles
  - Gaussian priors: x_1 prefers the center
    x_2, x_3, x_4 prefer to be straight
- observation: hand position $y = [y_1, y_2]$
- geometric arm simulation $y = g(x)$
  implicitly defines likelihood $p(y \mid x)$

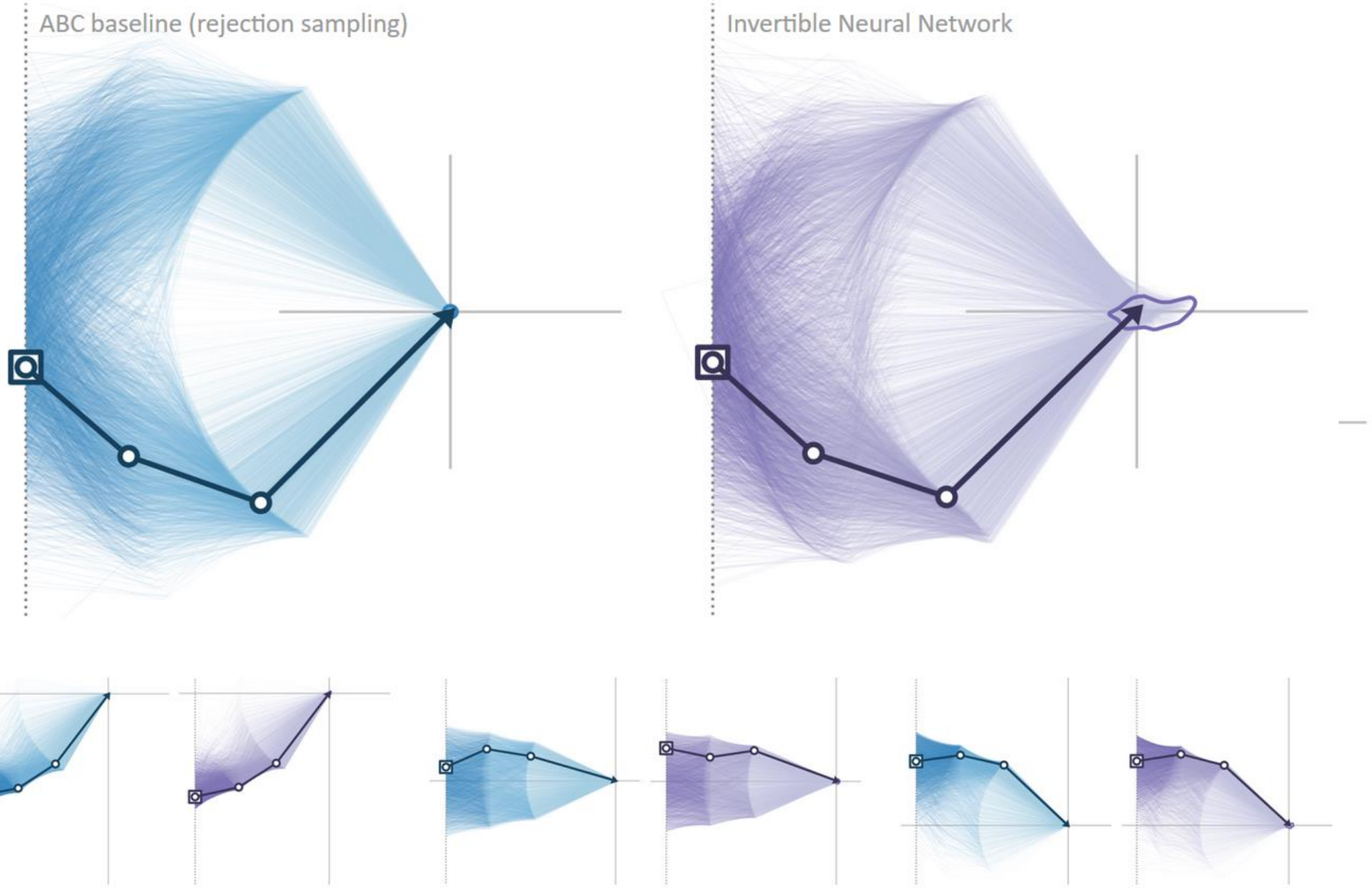
INN infers posterior $p(x \mid \hat{y})$ for given hand position $\hat{y}$

- For $t \in 1, \dots, T$:
  Sample $z^{(t)} \sim \mathcal{N}(0, \mathbb{I})$ and compute $x^{(t)} = f_\theta(\hat{y}, z^{(t)})$

posterior distribution $\mathrm{p}^*(x|\hat{y})$

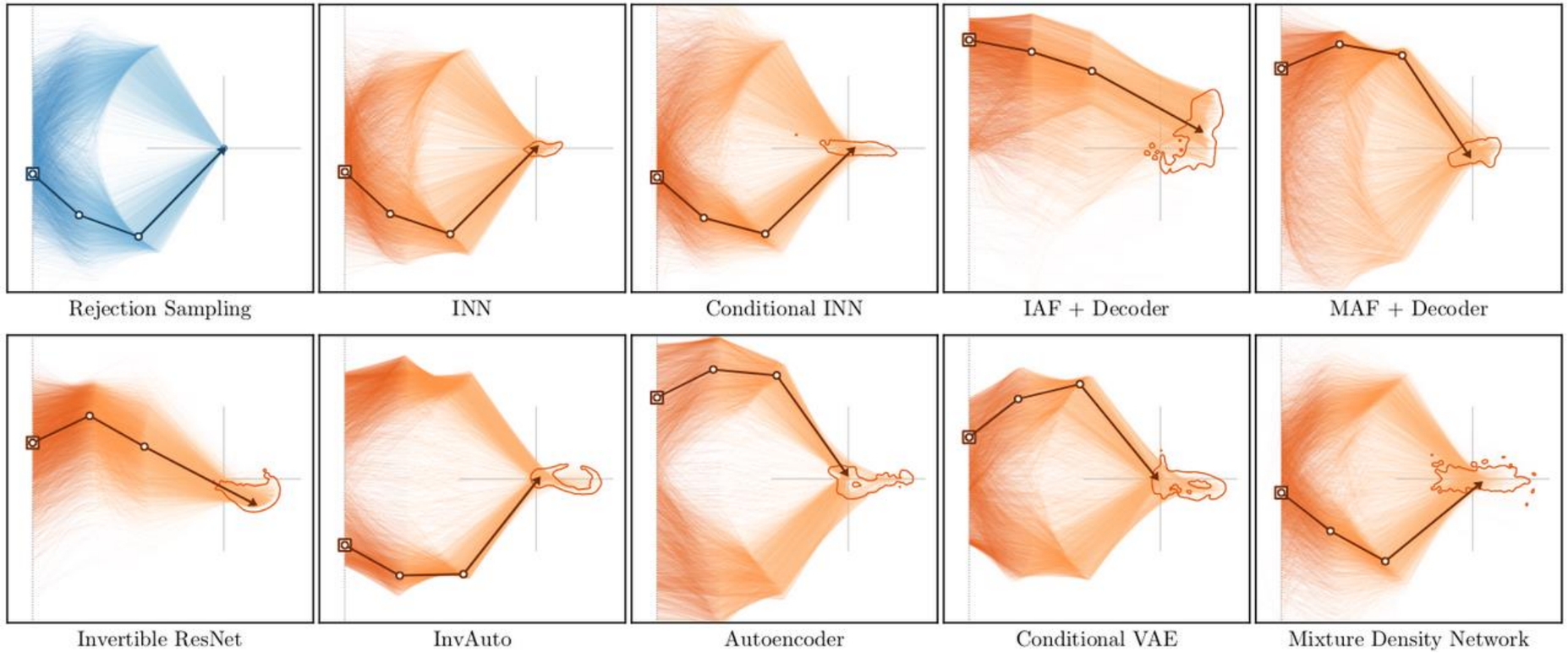


Non-linear Toy Example: Inverse Kinematics

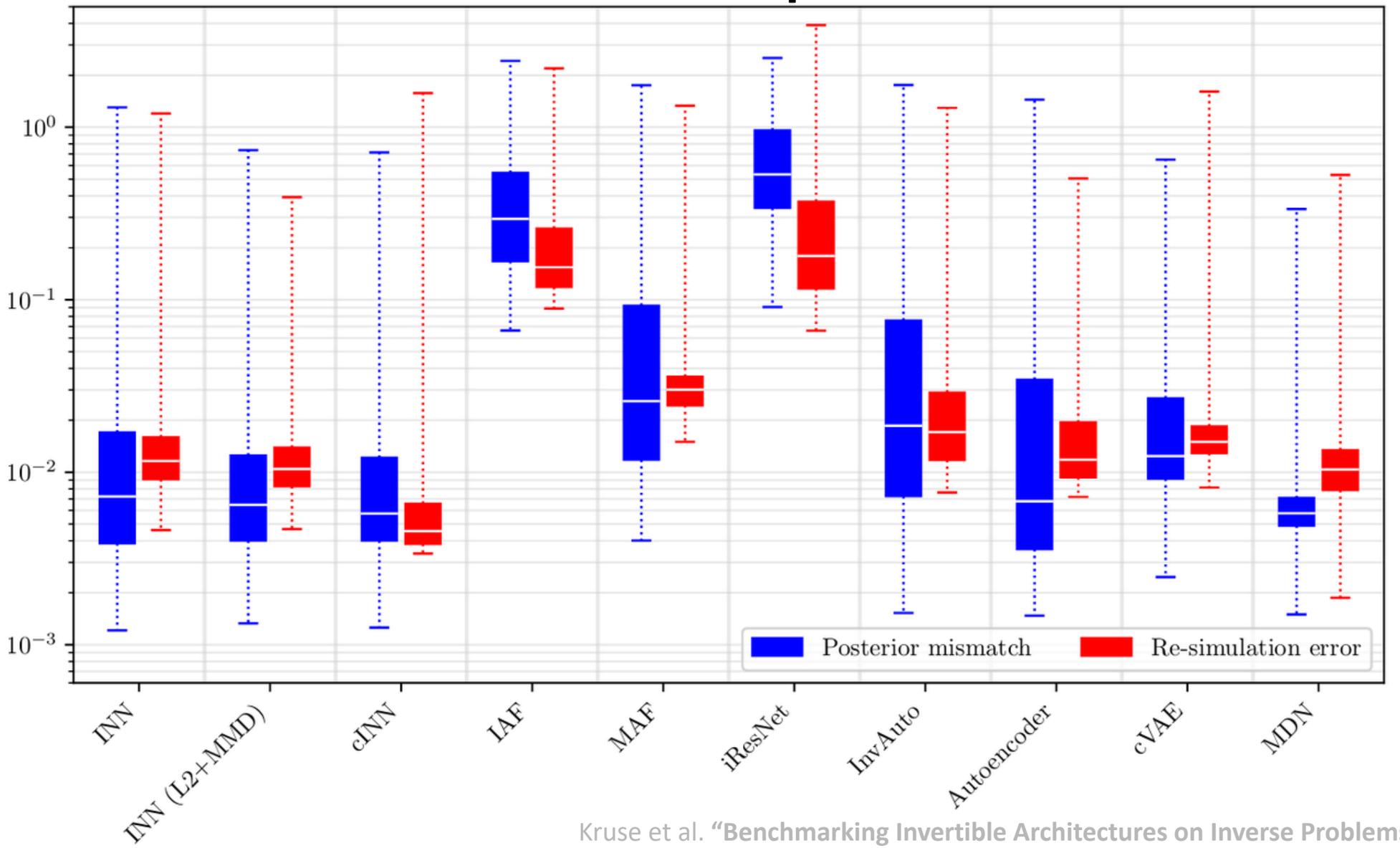


Comparison of Invertible Architectures
Kinematics Example

- Gound-truth posterior can be approximated with rejection sampling ("Approximate Bayesian Computation" – ABC)
- Performance metrics:
  - MMD between estimated and ground-truth posterior
  - re-simulation error $\|s(\hat{x}) - \hat{y}\|_2^2$: where does $s(\hat{x})$ actually land?



Comparison of Invertible Architectures
Kinematics Example



Kruse et al. **"Benchmarking Invertible Architectures on Inverse Problems"**, INNF 2019.