Tutorial "Normalizing Flows" Part 2

Ullrich Köthe

Visual Learning Lab, Heidelberg University

CVPR 2021, June 2021

Recap: How do normalizing flows work? How do they differ from other generative models?

Ullrich Köthe
Visual Learning Lab, Heidelberg University

Tutorial „Normalizing Flows“ at CVPR 2021
June 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

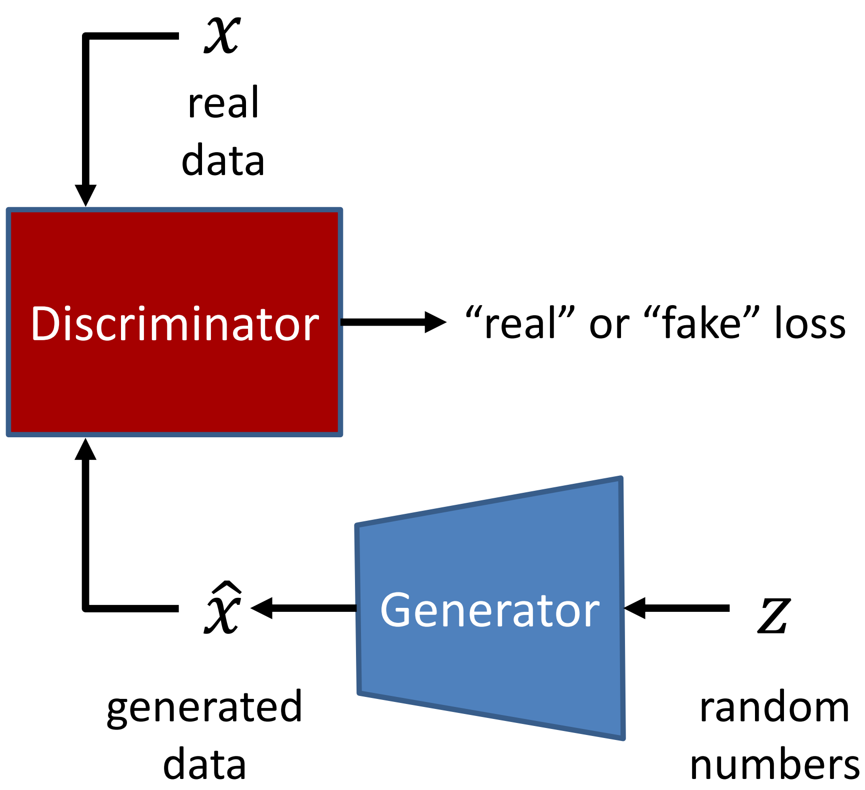
$$p(y \mid x) = \frac{p(x \mid y)\, p(y)}{p(x)}$$

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

Generative Modelling as a Basis for Interpretable Deep Learning

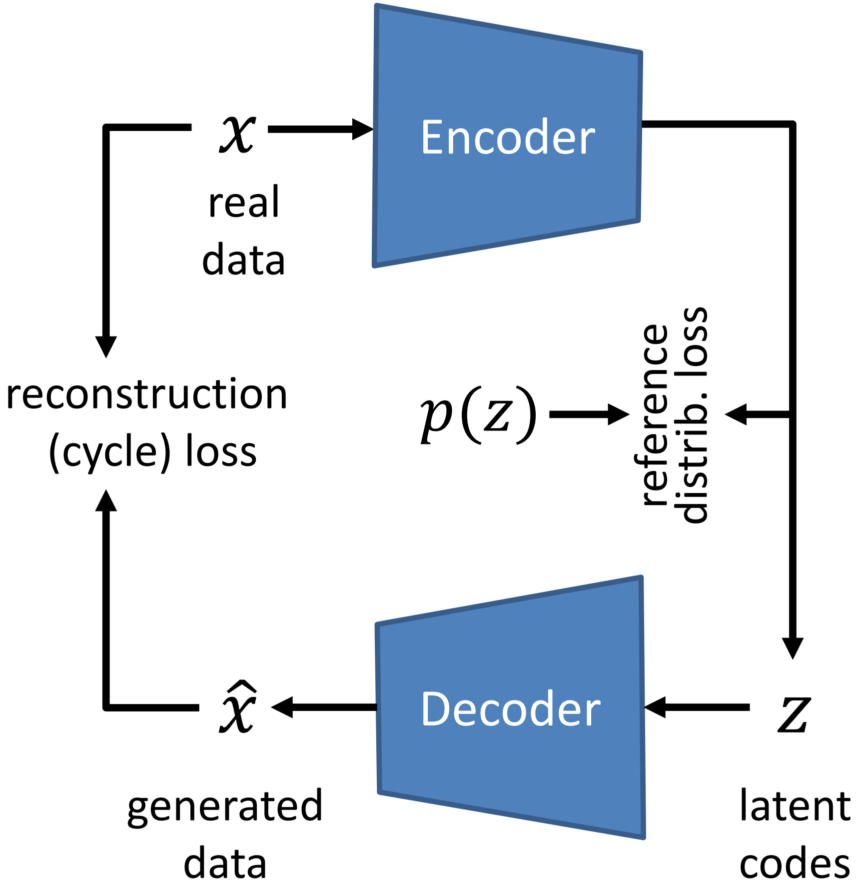
GANs

(Generative Adversarial Networks)



generation only

(Variational) Autoencoders



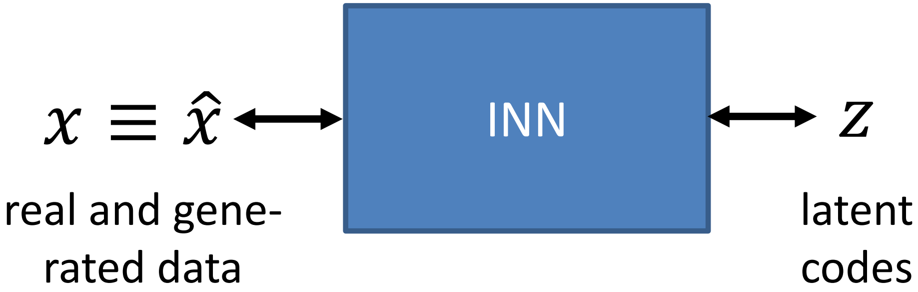
lossy encoding / decoding

Normalizing Flows

(Invertible Neural Networks, INNs)

maximum likelihood loss

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



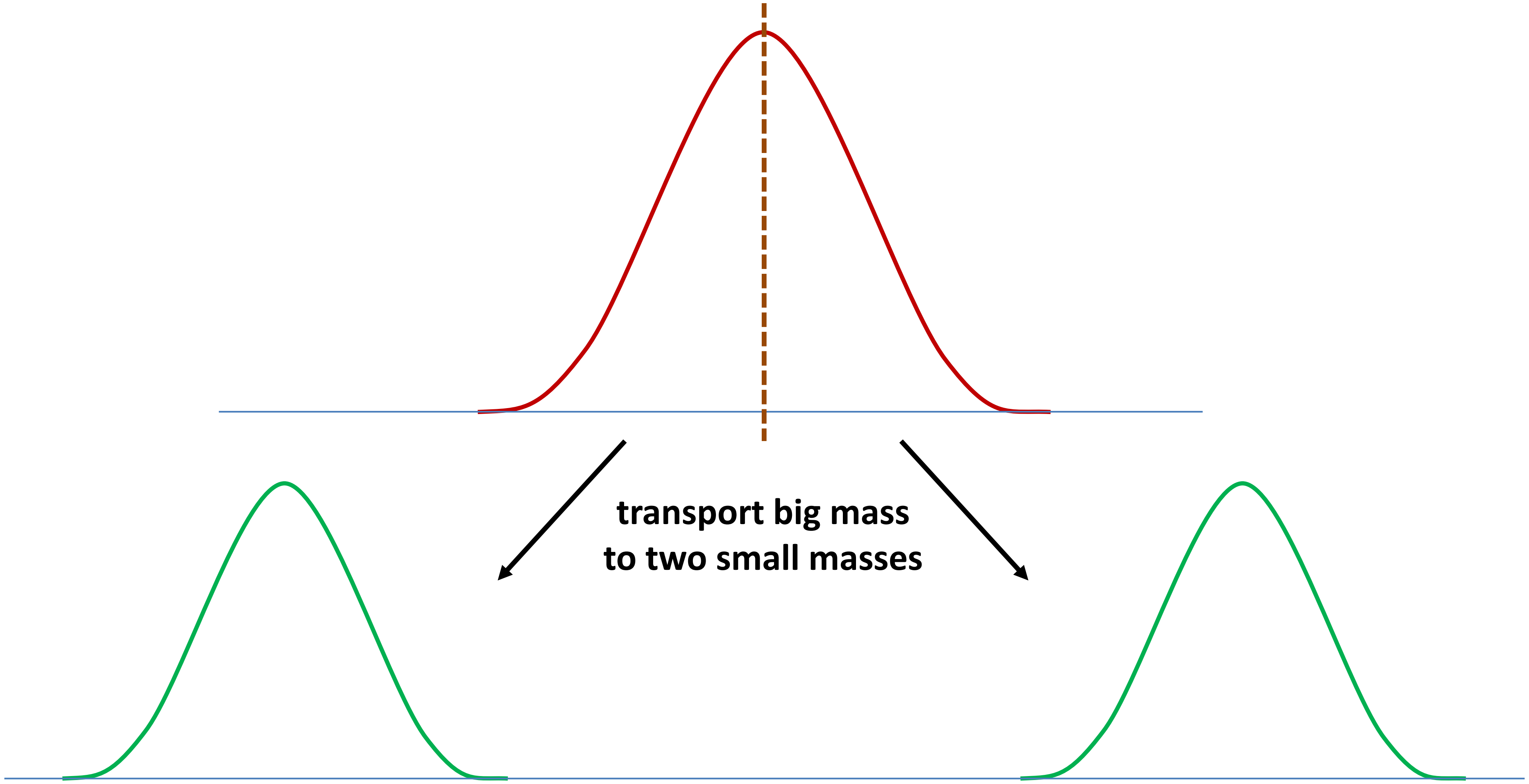
Encoder and Decoder the are same network, run forward / backward

lossless encoding / decoding

Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

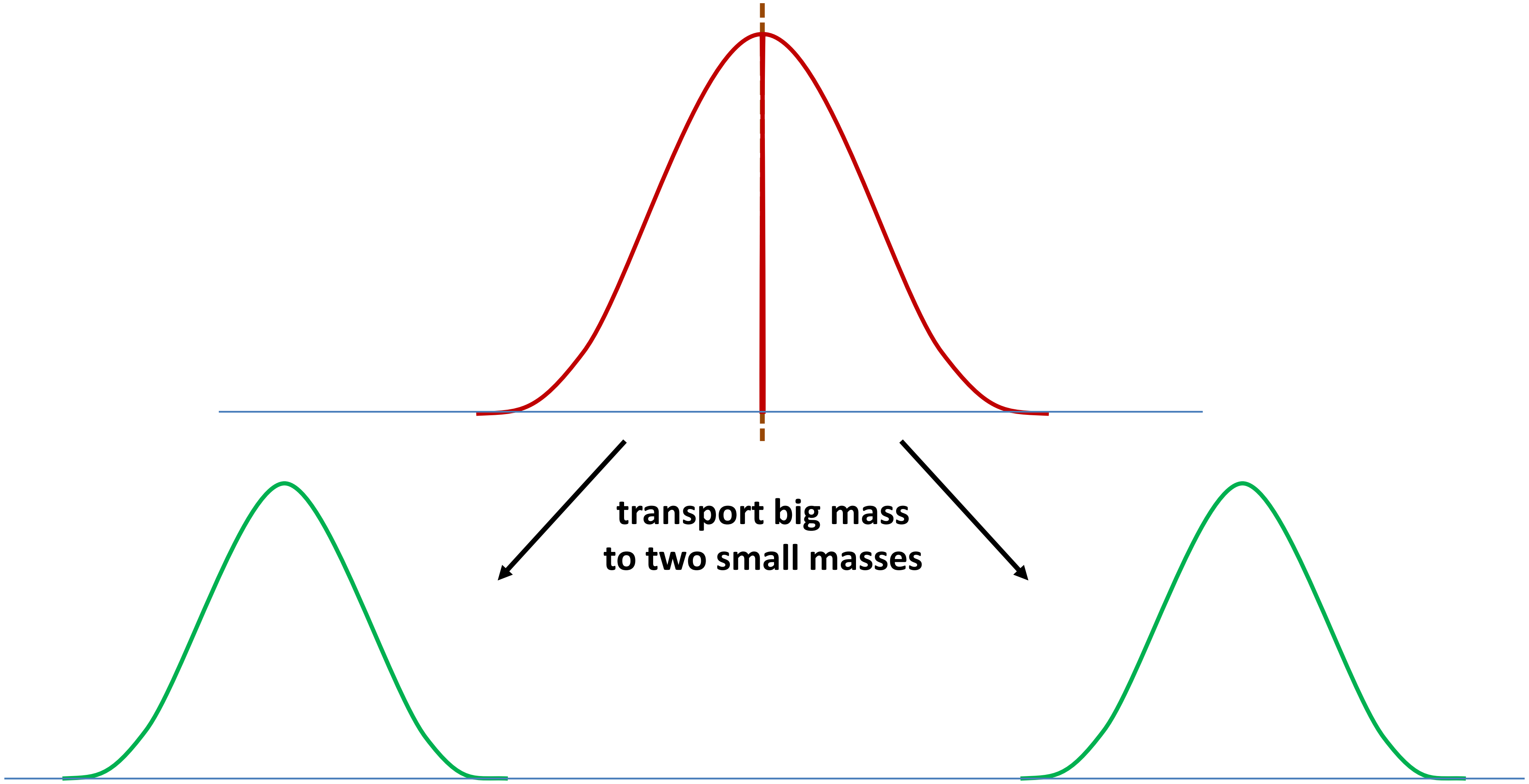
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

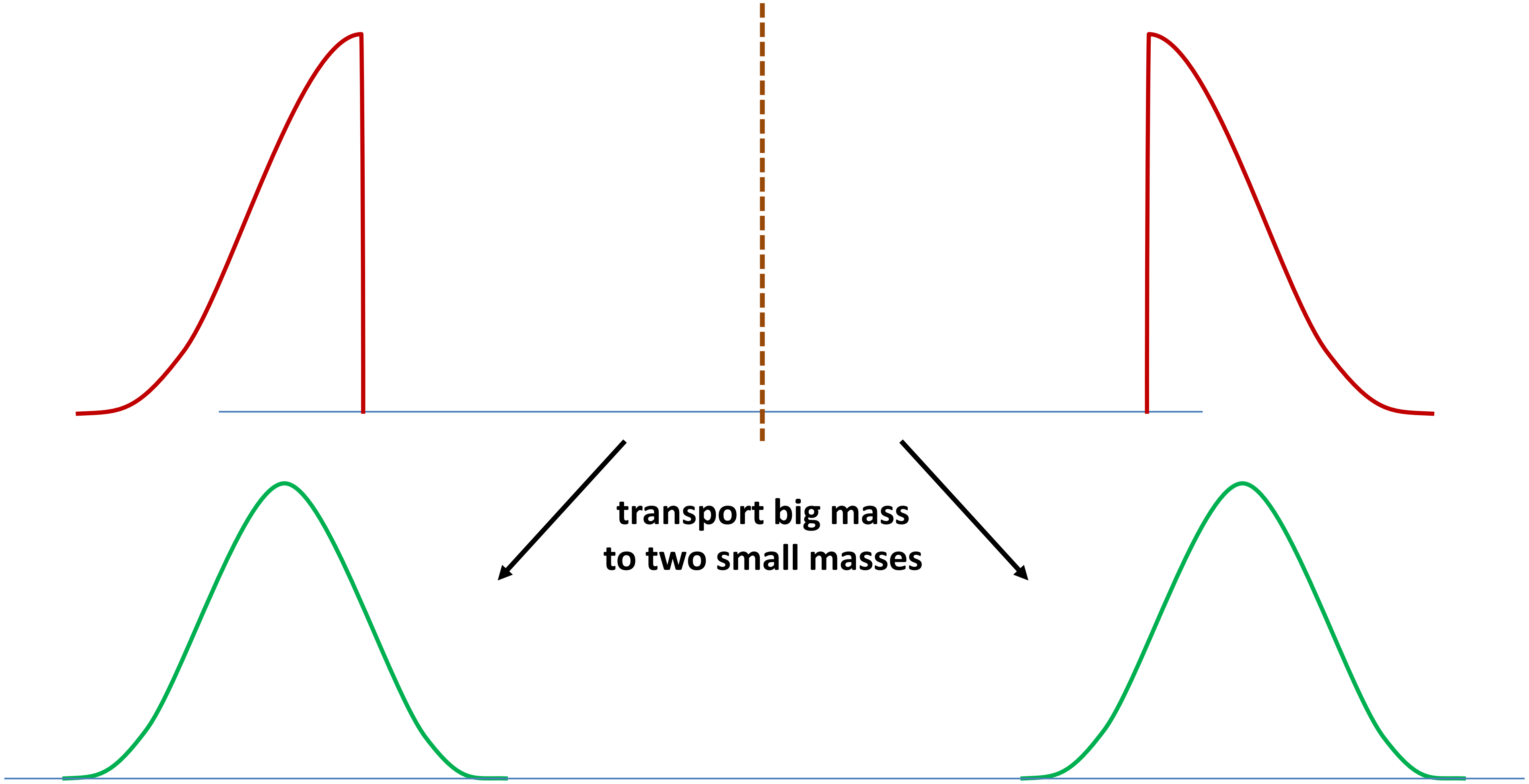
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

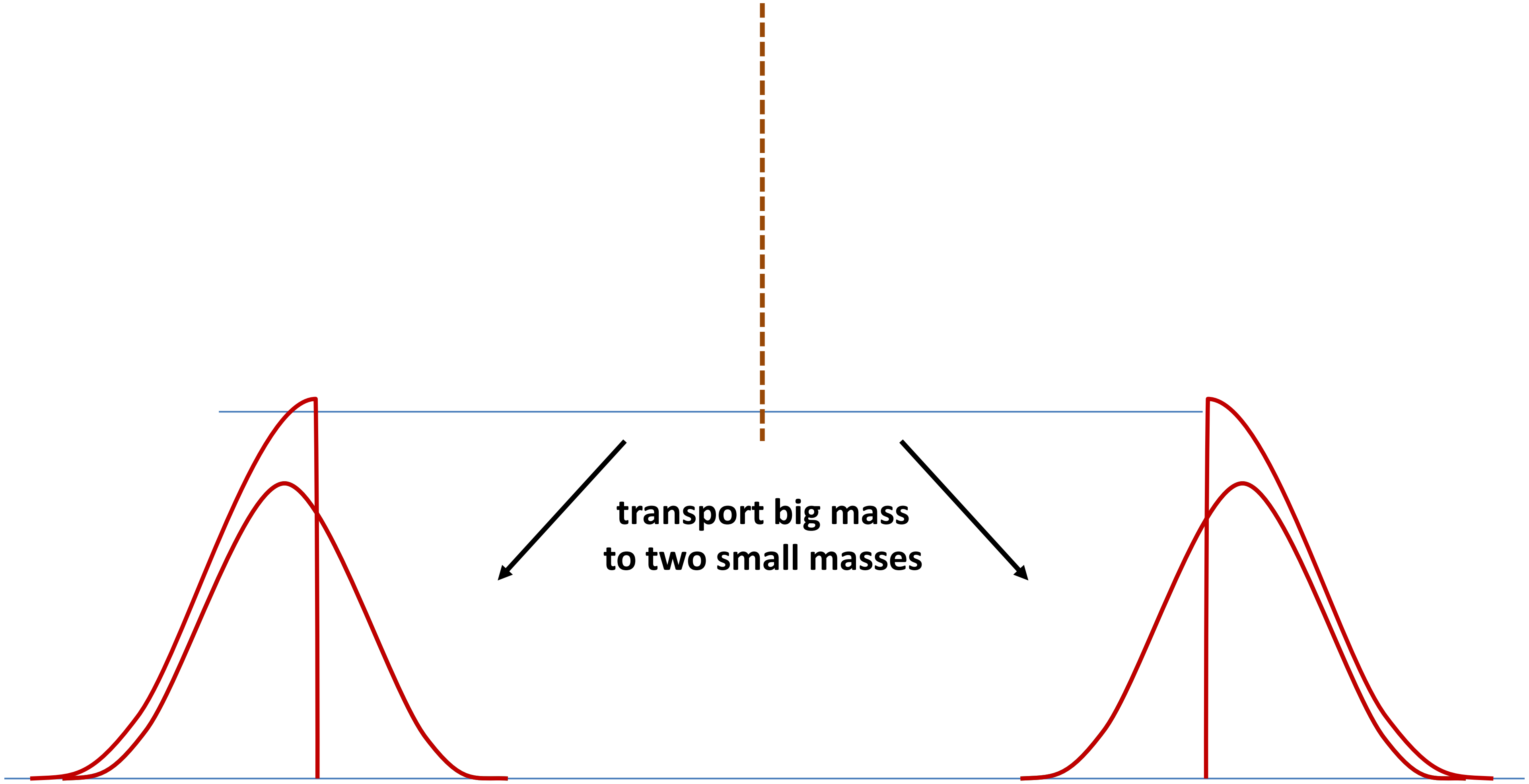
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

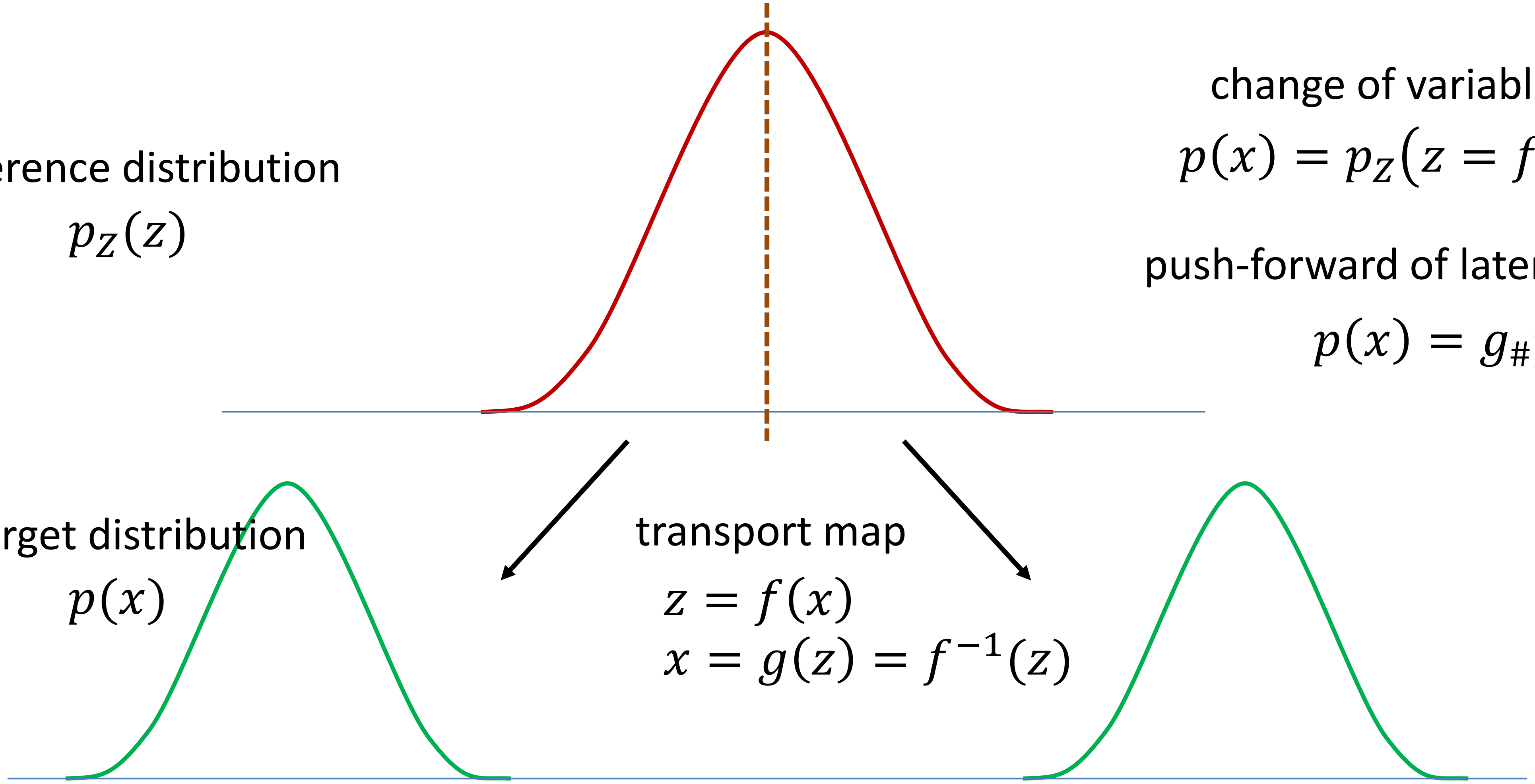
- Example: transport (“flow”) from simple “sand pile” to target „sand piles“



Normalizing flows

Model complicated probabilities as bijective mappings of simple ones

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



reference distribution

$p_Z(z)$

target distribution

$p(x)$

transport map

$z = f(x)$

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

change of variables formula

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

push-forward of latent distribution

$$p(x) = g_\# p_Z(z)$$

Multiple Possibilities for Normalizing Flows

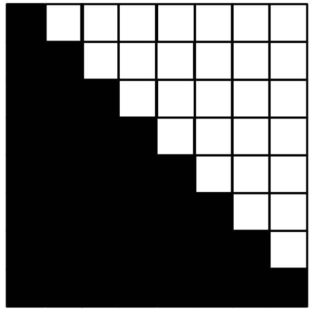
Autoregressive Models

Chain rule decomposition:

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

triangular reparameterization:

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



inverse direction inefficient

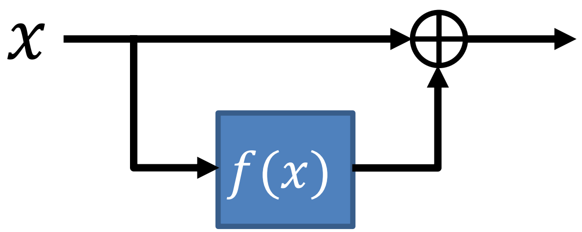
⇨ use two complementary nets

example: parallel WaveNet

iResNets

(invertible residual networks)

Residual block:



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

is invertible when

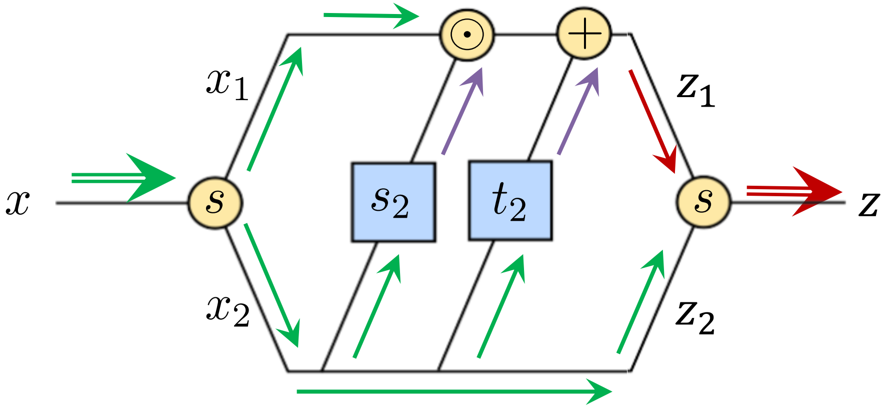
$$\|f(x)\|_{\text{Lipshitz}} < 1$$

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

How do you make ResNets invertible and why would you care?

Ullrich Köthe
Visual Learning Lab, Heidelberg University

Tutorial „Normalizing Flows“ at CVPR 2021
June 2021

Recap: What is a ResNet?

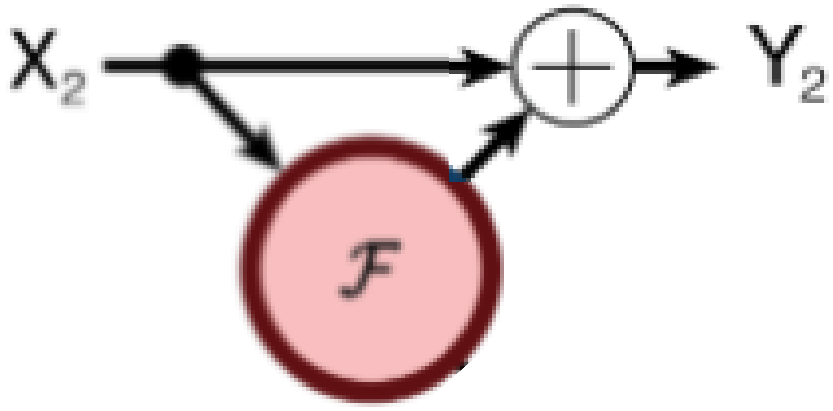
- Instead of modeling the transition from layer l to $l+1$

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

  model the difference (residual) between consecutive layers

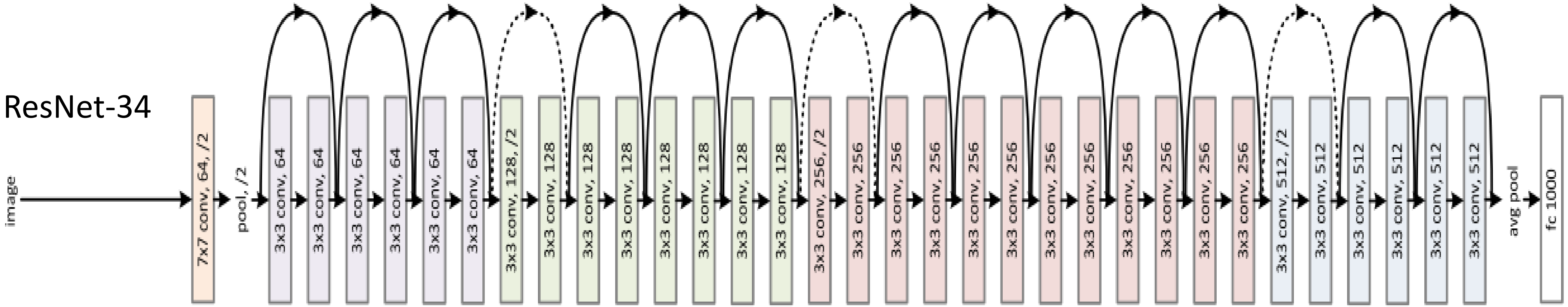
$$z_{l+1} - z_l = \mathcal{F}_l(z_l) \quad \Leftrightarrow \quad z_{l+1} = z_l + \mathcal{F}_l(z_l)$$

  - 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

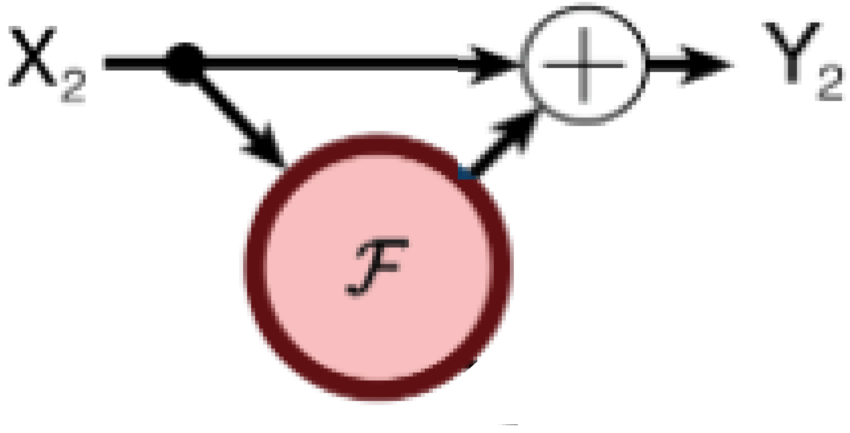


He et al. **"Deep residual learning for image recognition",** CVPR 2016.

RevNets: Memory-efficient backpropagation

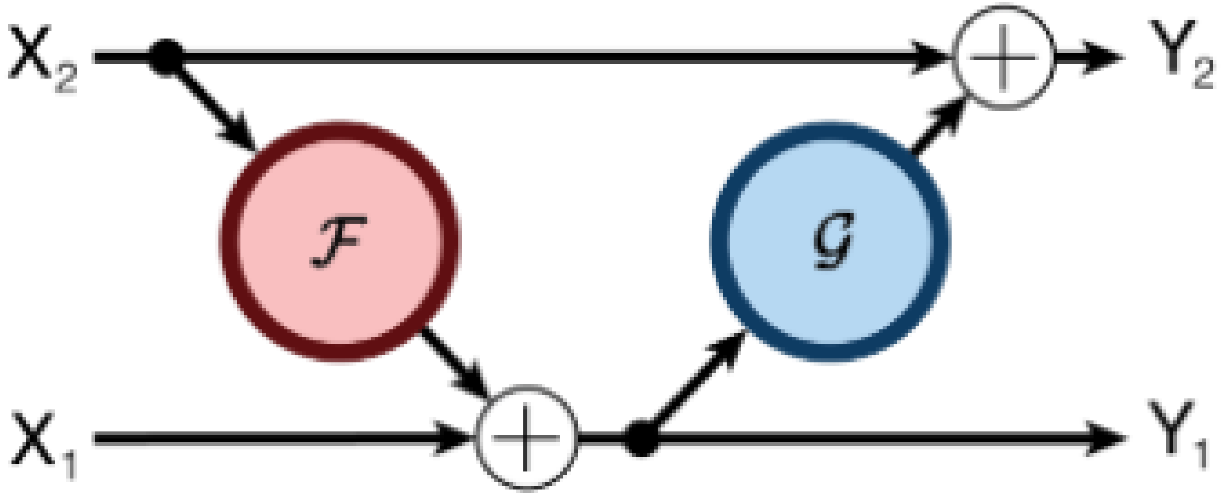
- Simple application of coupling layers: replace residual blocks with coupling blocks
  - Do not store activations during the forward pass of training
  - Recompute them on the fly during backpropagation, using the invertible architecture

Residual block



becomes: ⇨

Coupling block



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

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

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

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

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

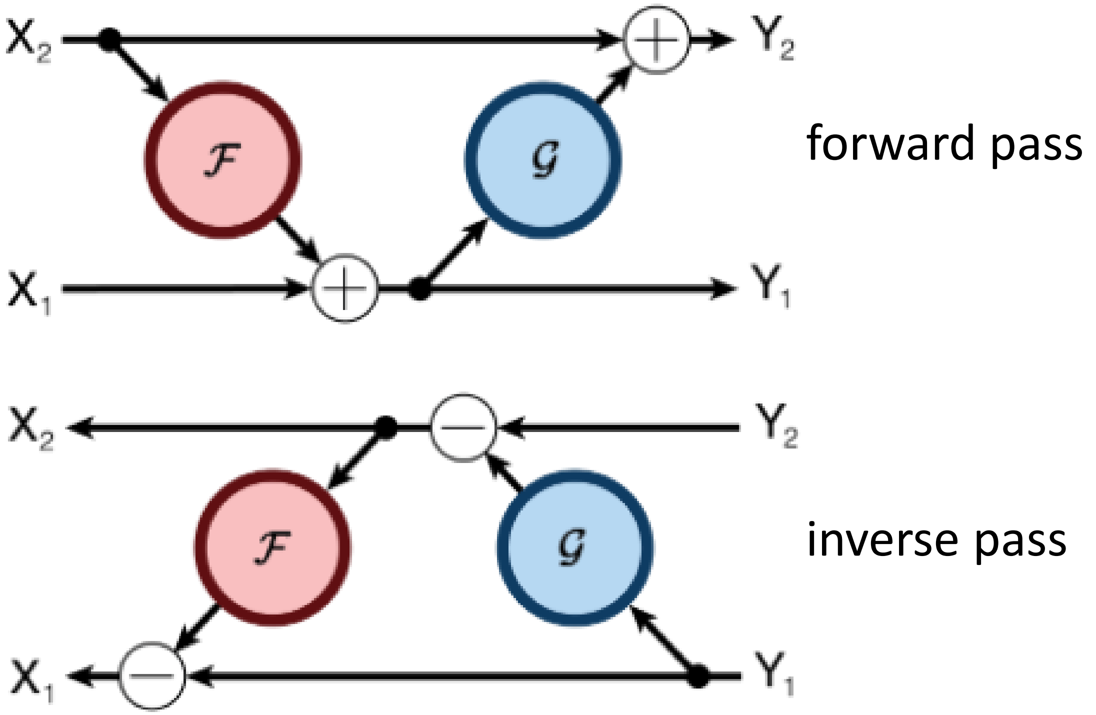
$\overline{w}_{\mathcal{F}} \leftarrow \left(\frac{\partial \mathcal{F}}{\partial w_{\mathcal{F}}}\right)^{\top} \overline{z}_1$

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

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

end function

"isolated" autodiffs of G and F



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

RevNets: Memory-efficient backpropagation

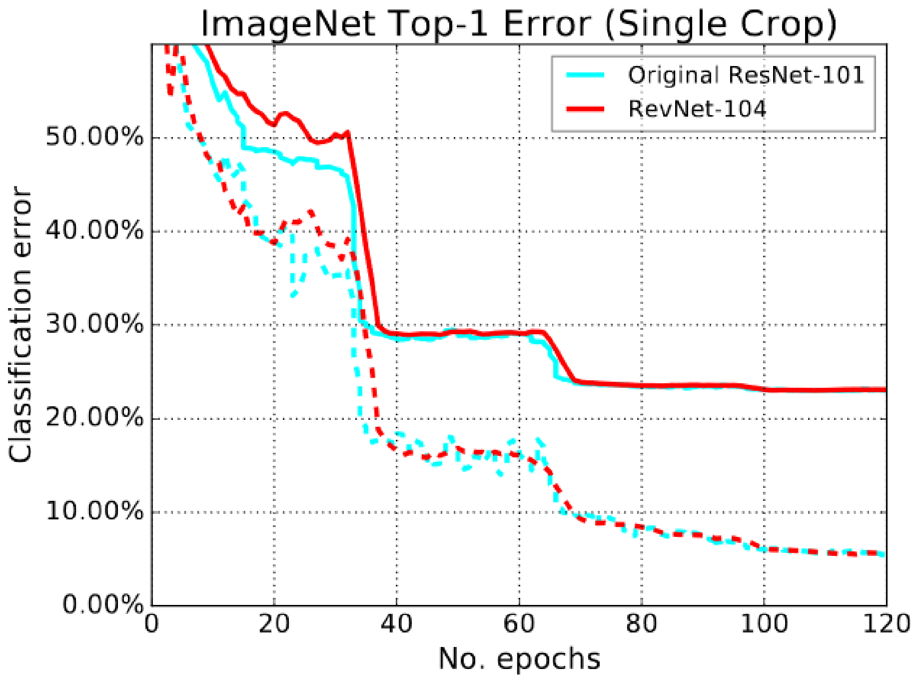
- Performance example: ResNet-101 vs. RevNet-104 on ImageNet

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

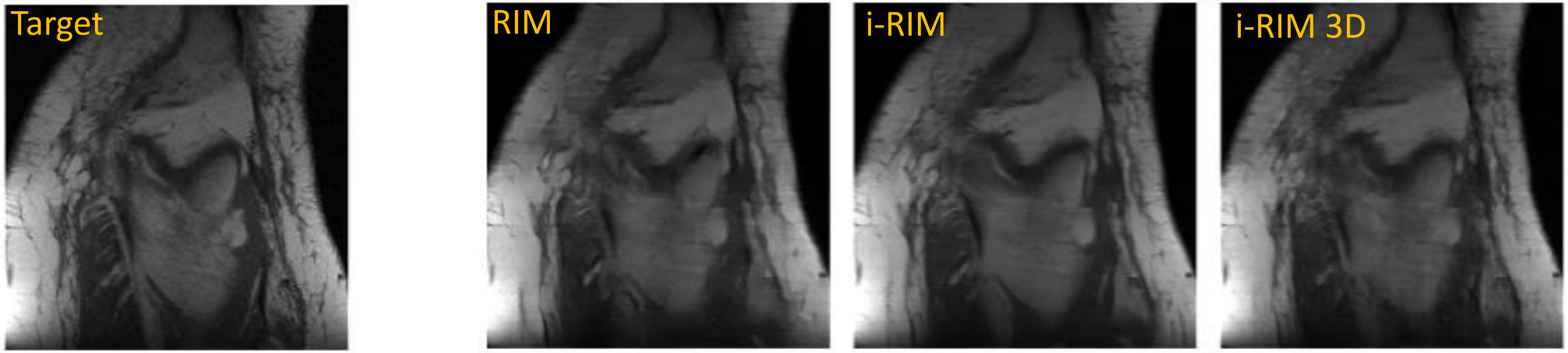


Table 1: Comparison of memory consumption during training and testing.

	RIM	i-RIM 2D	i-RIM 3D
Size Machine State (η, 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 (η, 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

Putzky & Welling **"Invert to Learn to Invert",** NeurIPS 2019.

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 iterations:

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

backward pass

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

Making ResNets Invertible: i-ResNets and Residual Flows

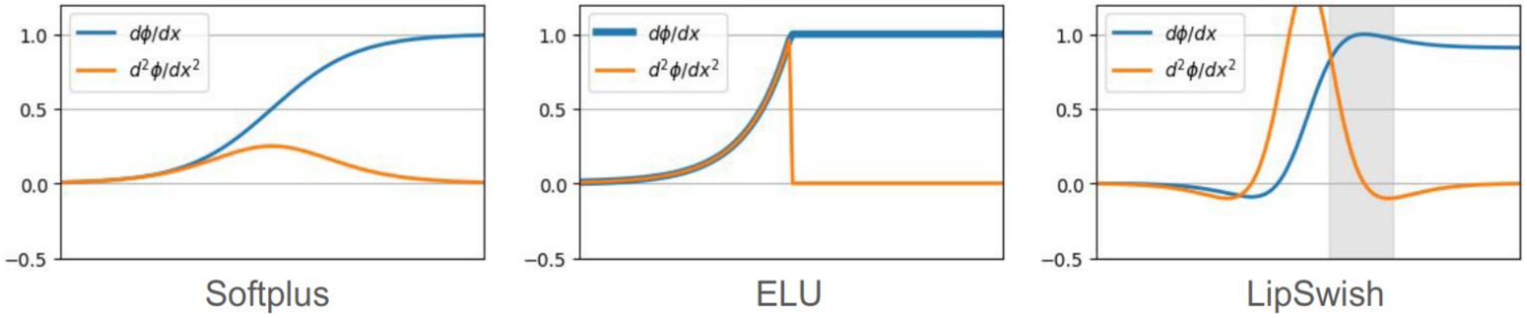
- How to achieve the Lipschitz bound?
  - Concatenation is Lipschitz, when each transition is so
  - Linear/convolutional layers: **normalize weight matrices** with $c < 1$ and largest singular value $\tilde{\sigma}_i \leq \|W_i\|_2$ estimated by (one iteration of) power method

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

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

  - Activation function: $\forall x$: $|\phi'(r)| \leq 1$ is 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 $\boldsymbol{\phi(r) = \mathrm{LipSwish}(r) = 0.909\, r/(1 + \exp(-\beta\, r))}$**



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

- How to perform maximum likelihood training?
  - Need the gradient of the log-determinant of the Jacobian
  - Approximate via truncated power series or unbiased log density estimator

$$\ln\left|\det\left(I + J_{g_{\theta_t}}(\mathbf{x}_t)\right)\right| \approx \sum_{k=1}^{n}(-1)^{k+1}\frac{\operatorname{tr}\left(J_{g_{\theta_t}}(\mathbf{x}_t)^k\right)}{k} \approx \mathbb{E}_{n,v}\left[\sum_{k=1}^{n}\frac{(-1)^{k+1}}{k}\frac{v^T[J_g(x)]^k v}{\mathbb{P}(N\geq k)}\right]$$

  - Very recent new possibility: use relative gradient (i.e. multiplicative instead of additive perturbation)

  ⇨ Gradient update calculation reduces to matrix-vector products (try on your own risk :-)

$$W_t \leftarrow W_t + \gamma(\mathbf{x}_{t-1}(\delta_t^T W_t^T) + \mathbb{I})W_t$$

Behrmann et al. **“Invertible Residual Networks”**. ICML 2019. Chen et al. **“Residual Flows for Invertible Generative Modelling”,** NeurIPS 2019.
Gresele et al. **“Relative gradient optimization of the Jacobian term”**, INNF+ 2020.

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**

RealNVP: Invertibility vie Coupling Layers

Ullrich Köthe
Visual Learning Lab, Heidelberg University

Tutorial „Normalizing Flows“ at CVPR 2021
June 2021

Invertible Neural Networks (INNs) with Coupling Layers

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

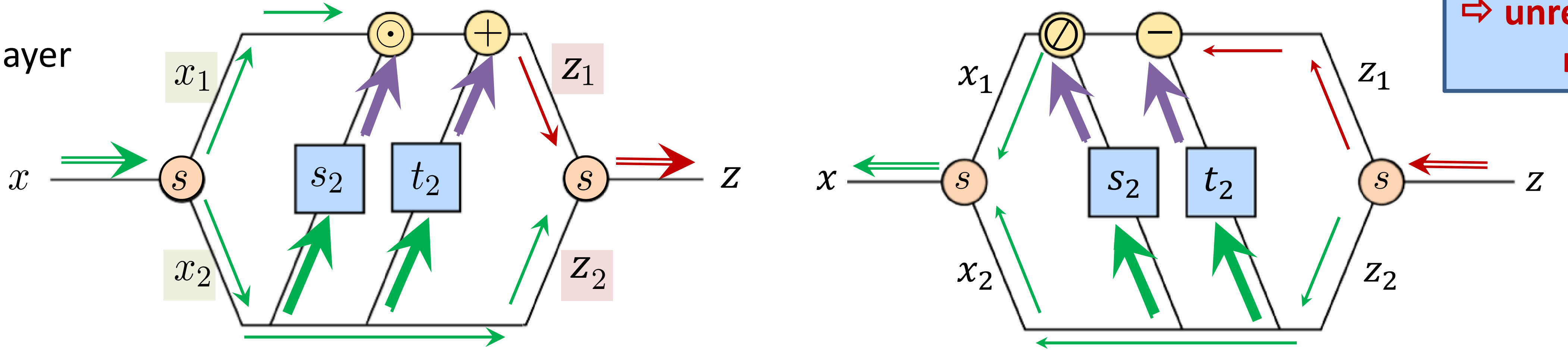
- Network is a sequence of *affine coupling layers*
- Each coupling layer splits its input $x \in \mathbb{R}^D$ into two halves $x_1, x_2 \in \mathbb{R}^{D/2}$
- Upper half is subjected to an affine transformation ⇨ outputs $z_1, z_2 \in \mathbb{R}^{D/2}$
- Affine coefficients are computed by standard 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

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

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

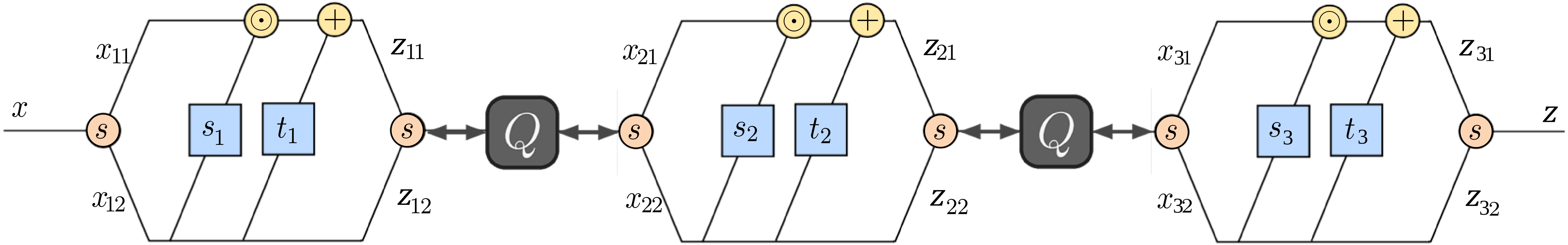
Inverse computation: $x_1 = (z_1 - t_2(z_2)) \oslash s_2(z_2), \qquad x_2 = z_2$

Coupling layer

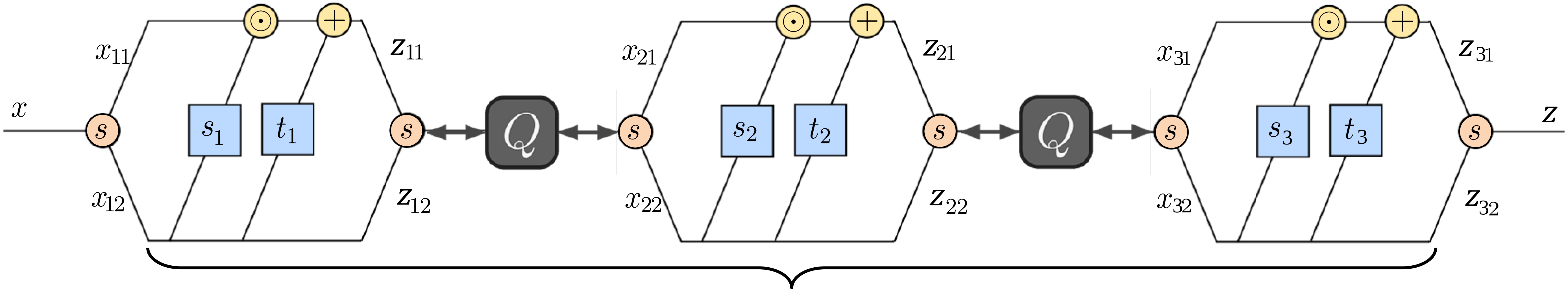


Deep INNs

- 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



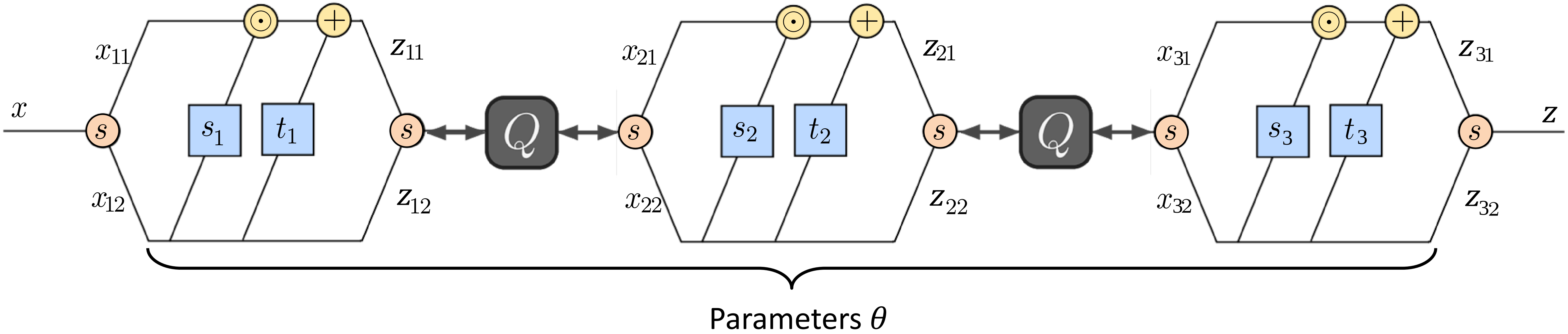
Tractable data likelihood via change-of variables formula: $p_\theta(x) = p_Z(z = f_\theta(x)) \cdot |\det \nabla f_\theta(x)|$

⇨ Negative log-likelihood has especially simple form when $p_Z(z)$ is standard normal

$$-\log p_\theta(x) = -\log p_Z\big(z = f_\theta(x)\big) - \log |\det \nabla f_\theta(x)|$$
$$= \frac{D}{2}\log 2\pi + \frac{1}{2}\|f_\theta(x)\|_2^2 - \sum_l \mathrm{sum}\big(\log s_{\theta,l}(x_{l2})\big)$$

with $s_{\theta,l}(x_{l2})$ the multipliers at coupling layer l (note: $\log\det Q = 0$)

Training Deep INNs with Maximum Likelihood Loss



Negative log-likelihood: $-\log p_\theta(x) = \frac{D}{2}\log 2\pi + \frac{1}{2}\|f_\theta(x)\|_2^2 - \sum_l \mathrm{sum}\big(\log s_{\theta,l}(x_{l2})\big)$

⇨ Train by minimizing the NLL objective over training set $\{x^{(i)}\}_{i=1}^N$:

$$\hat{\theta} = \arg\max_\theta \; p_\theta\left(\{x^{(i)}\}_{i=1}^N\right) = \arg\max_\theta \prod_{i=1}^N p_\theta\left(x^{(i)}\right) = \arg\min_\theta \sum_{i=1}^N -\log p_\theta\left(x^{(i)}\right)$$

$$= \arg\min_\theta \sum_{i=1}^N \left(\frac{1}{2}\left\|f_\theta\left(x^{(i)}\right)\right\|_2^2 - \sum_l \mathrm{sum}\left(\log s_{\theta,l}\left(x_{l2}^{(i)}\right)\right)\right)$$

Conditional Modeling with INNs

- In practice, we often need to model conditionals $p(x \mid y)$ or $p(y \mid x)$ rather than $p(x)$

- Example: Generative classification
  - x are features, y are class labels
  - determine posterior $p(y \mid x)$ using Bayes rule

$$p(y \mid x) \sim p(x \mid y)\, p(y)$$

  ⇨ learn the likelihood $p(x \mid y)$ via a conditional INN (specifically, an IB-INN)

- Example: Solving inverse problems
  - x are hidden system parameters, y are observations of the system behavior
  - determine the posterior $p(x \mid y = \hat{y})$ to estimate parameters x from measured $\hat{y}$

  ⇨ learn $p(x \mid y)$ using synthetic data from a simulation $y = g(x; \mathrm{noise})$ of the forward process

INN Architectures for Conditional Inference

Split latent space

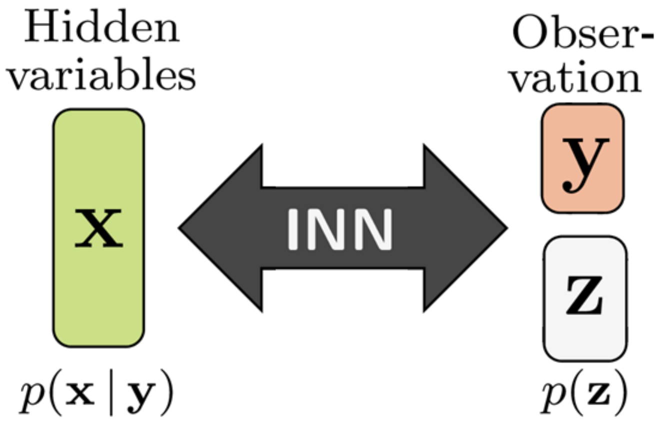
training: $(y, z) = f_\theta(x)$

s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$

compute $x = f_\theta^{-1}(\hat{y}, z)$

⇨ $x \sim p(x \mid \hat{y})$



historically first

Latent mixture INN

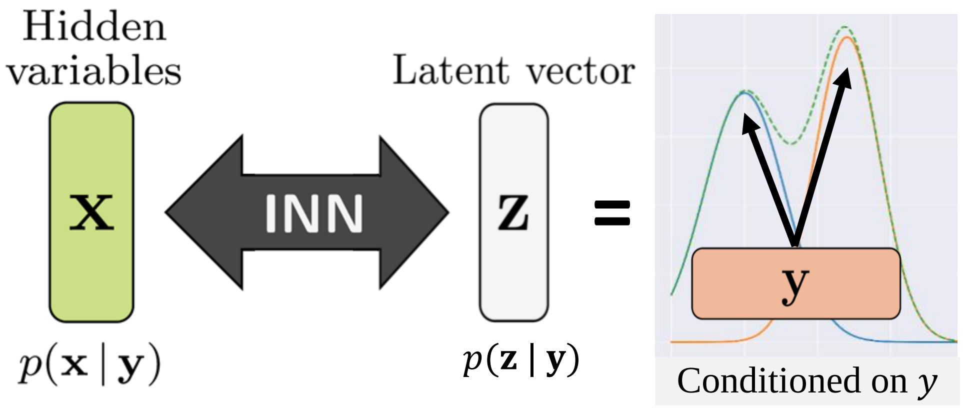
training: $z = f_\theta(x)$

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 $x = f_\theta^{-1}(z)$

⇨ $x \sim p(x \mid \hat{y})$



classification, disentanglement

Conditional INN

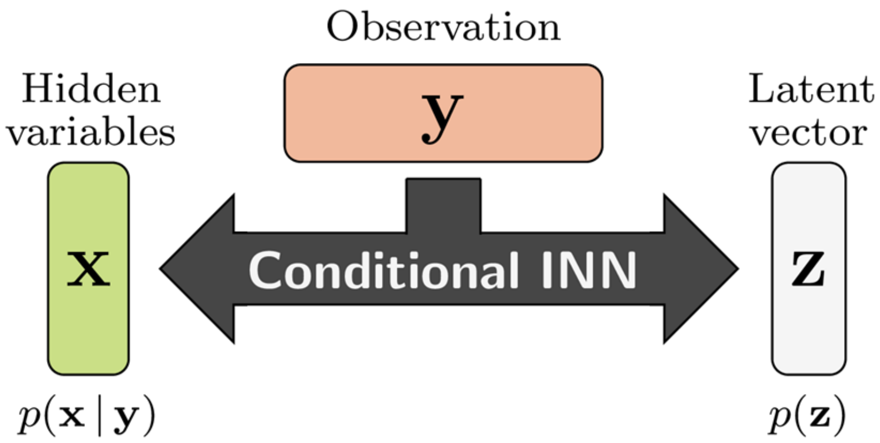
training: $z = f_\theta(x; y)$

s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$

compute $x = f_\theta^{-1}(z; \hat{y})$

⇨ $x \sim p(x \mid \hat{y})$



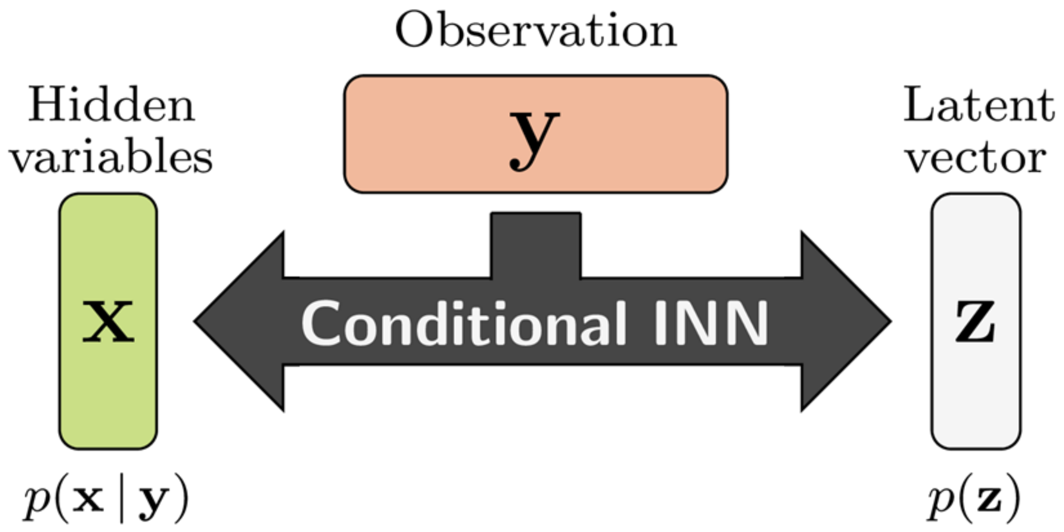
inverse inference

Conditional INN (cINN)

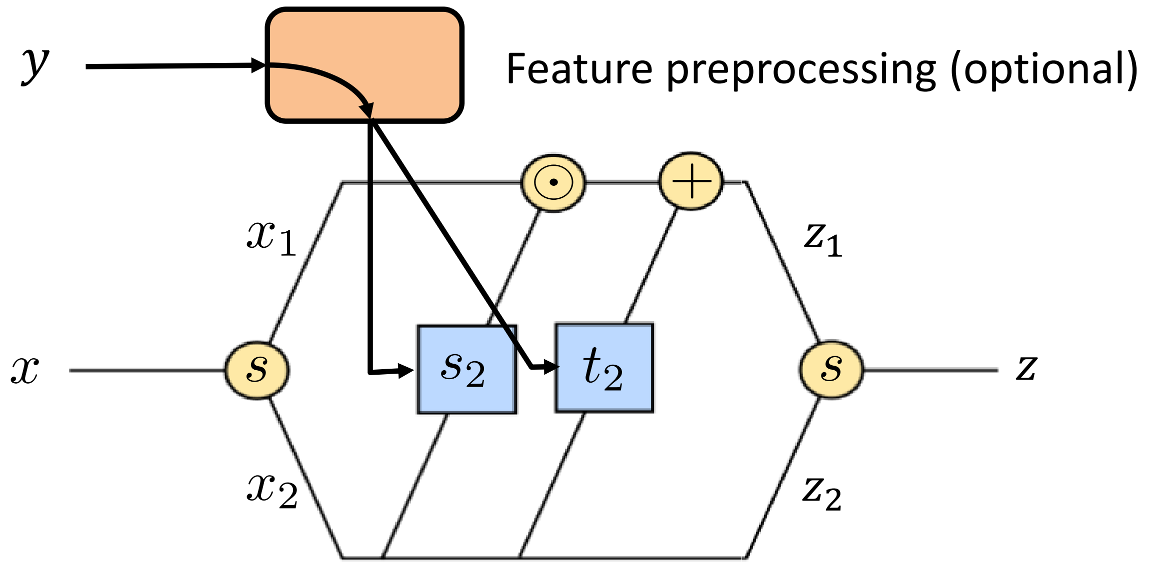
Conditional INN (cINN) adapts vanilla INN for conditional probabilities

- Reparametrize $x \sim p(x \mid y)$ as $x = g_\theta(z; y)$ with $z \sim p_Z(z)$ and forward process $z = f_\theta(x; y) = g_\theta^{-1}(x; y)$
- Minimum log-likelihood loss becomes

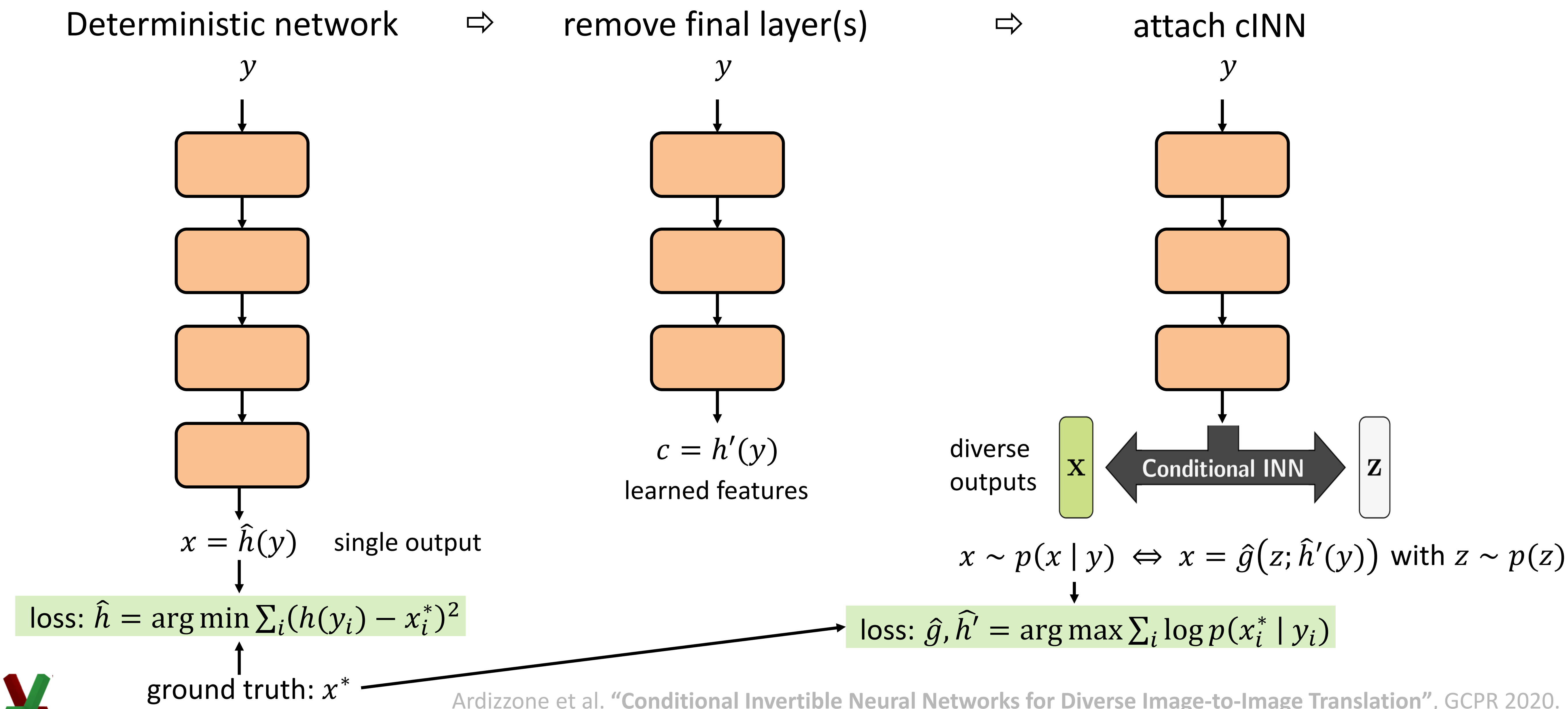
$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{N} \left(\frac{1}{2} \left\| f_\theta\left(x^{(i)}; y^{(i)}\right) \right\|_2^2 - \sum_l \operatorname{sum}\left(\log s_{\theta,l}\left(x_{l2}^{(i)};\ y^{(i)} \right) \right) \right)$$



simple change of coupling layer architecture:
feed y as additional input to subnets s, t



cINNs Turn Deterministic Networks into Probabilistic Ones



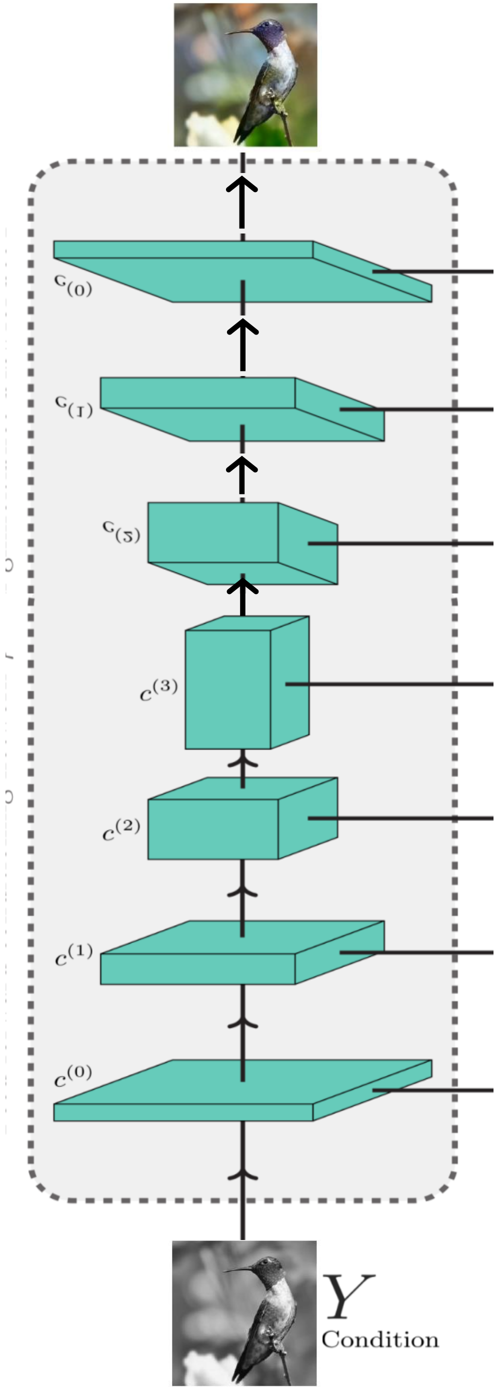
Ardizzone et al. **"Conditional Invertible Neural Networks for Diverse Image-to-Image Translation"**, GCPR 2020.

Example: Image-to-Image Translation

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y

$[y, \hat{x}]$

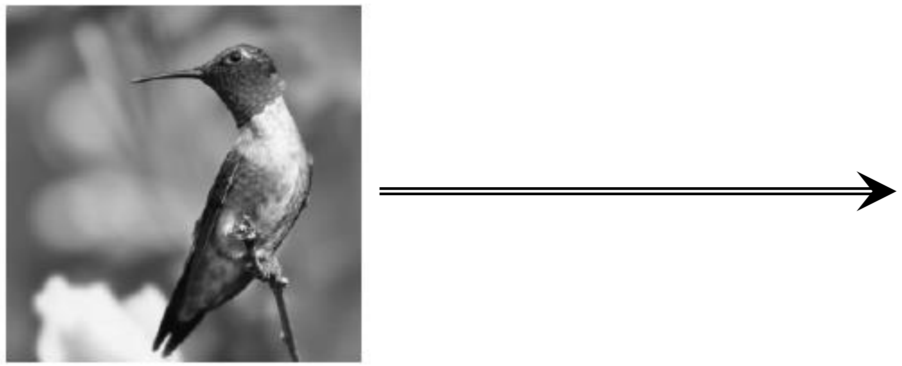


deterministic U-net

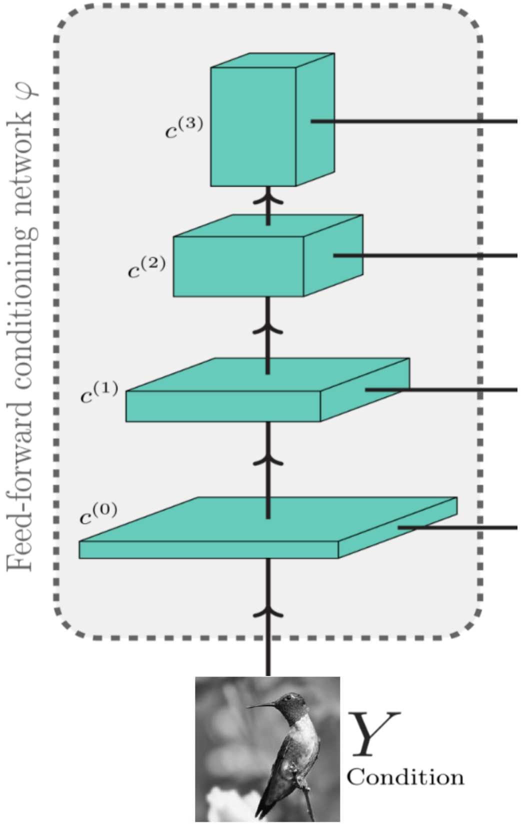
  - deterministic network: single result

Example: Image-to-Image Translation

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$



y

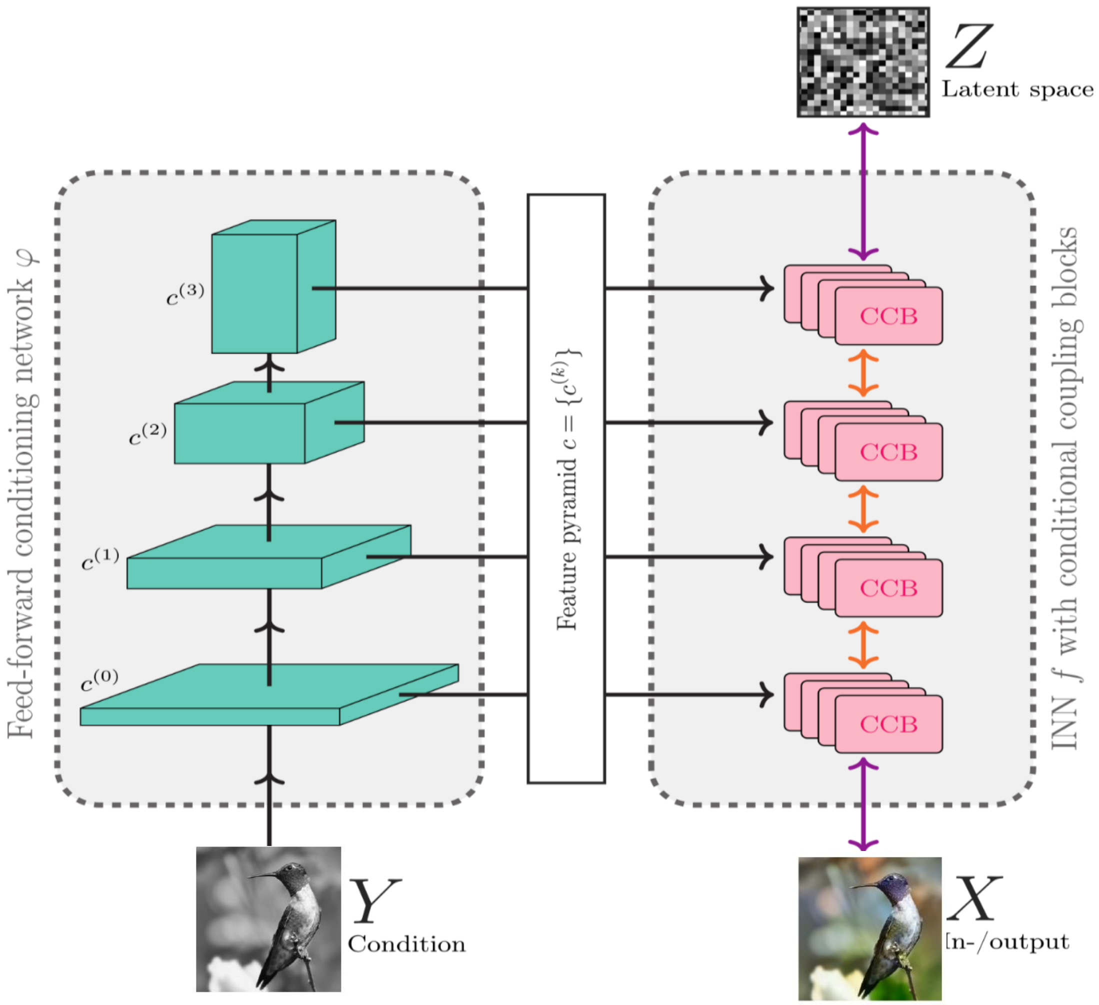


Ardizzone et al. **"Conditional Invertible Neural Networks for Diverse Image-to-Image Translation"**, GCPR 2020.

Example: Image-to-Image Translation

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y $\Longrightarrow$ $p(x|y)$



  - cINN: diverse results

Example: Image-to-Image Translation

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y

$z \sim p_Z(z)$

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

  - cINN: diverse results
  - Quiz: Which color image is the ground-truth?

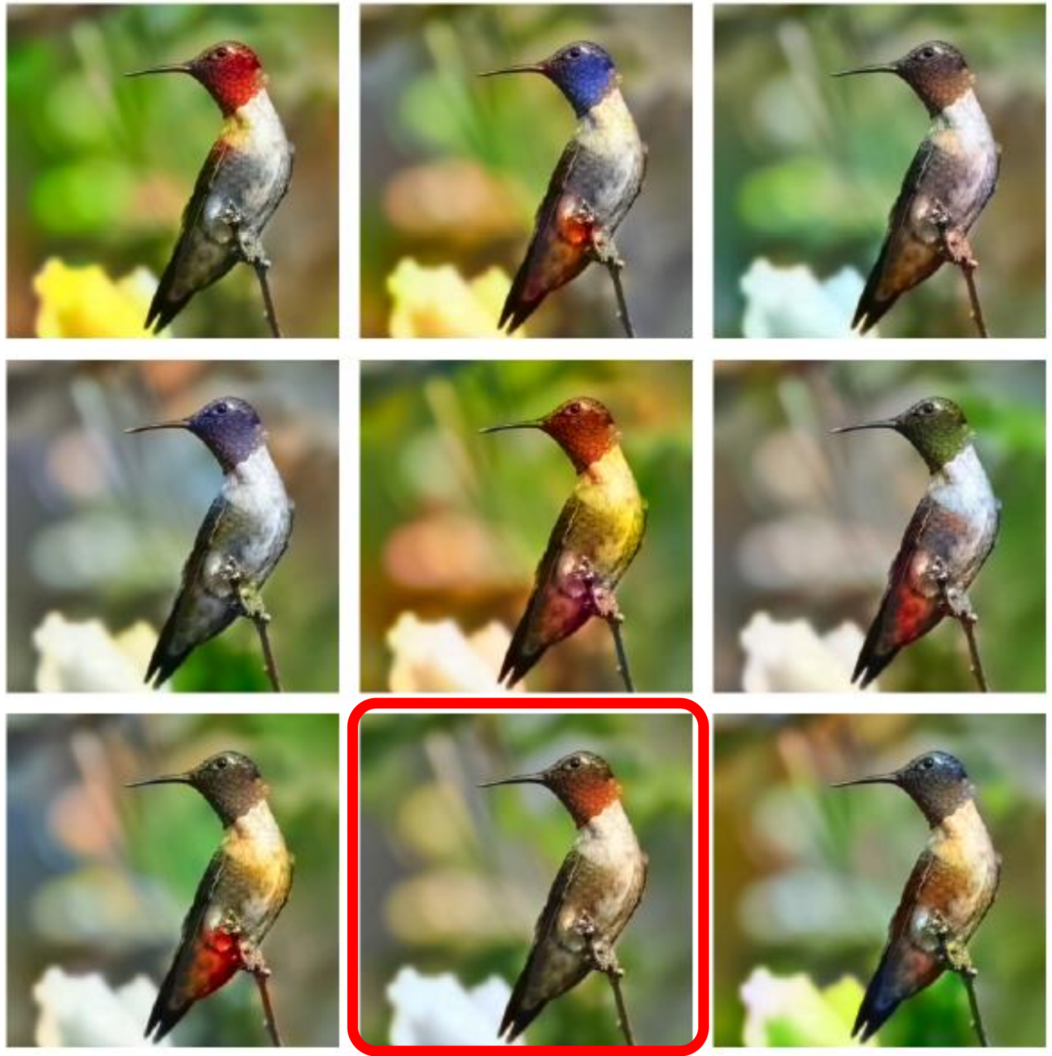
Example: Image-to-Image Translation

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y

$z \sim p_z(z)$

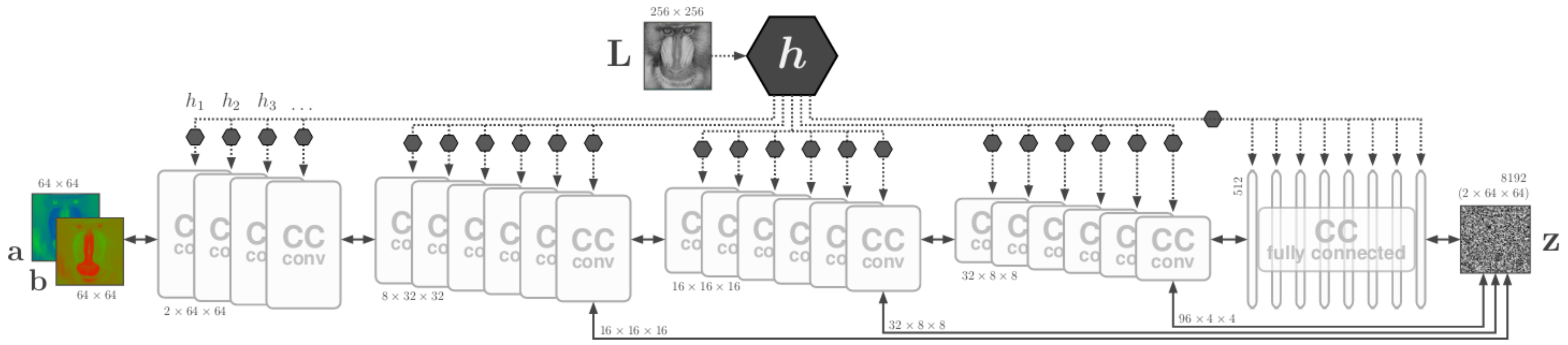
$x \sim \hat{p}(x \mid y)$



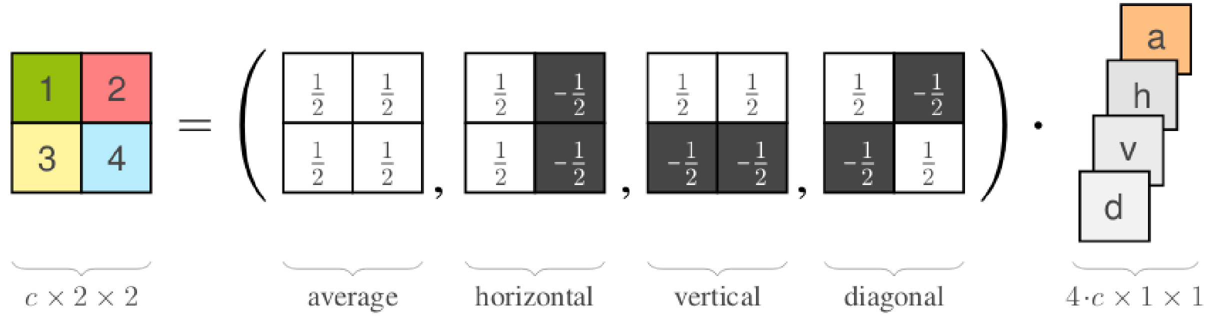
- cINN: diverse results
- Quiz: Which color image is the ground-truth?

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

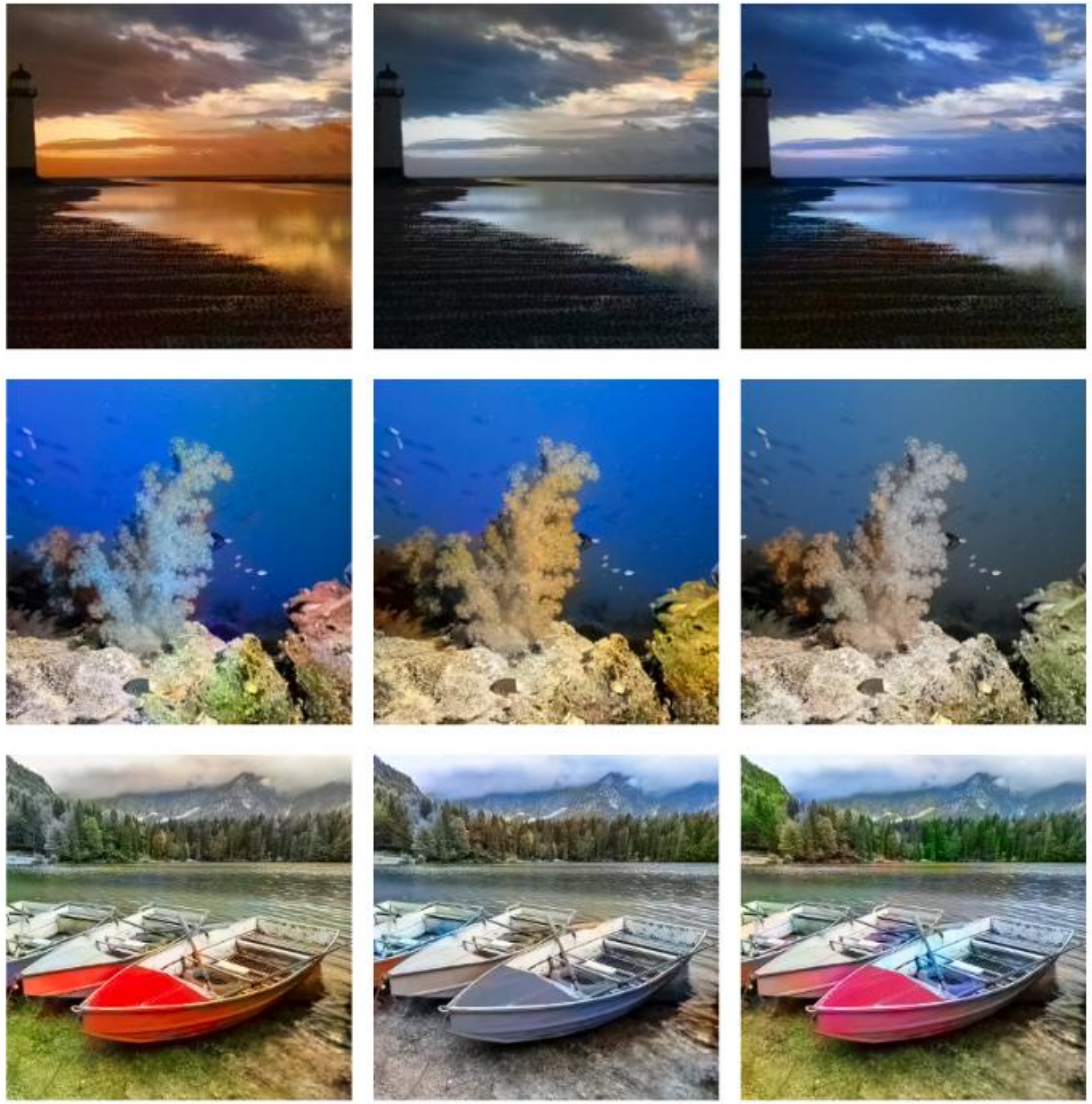


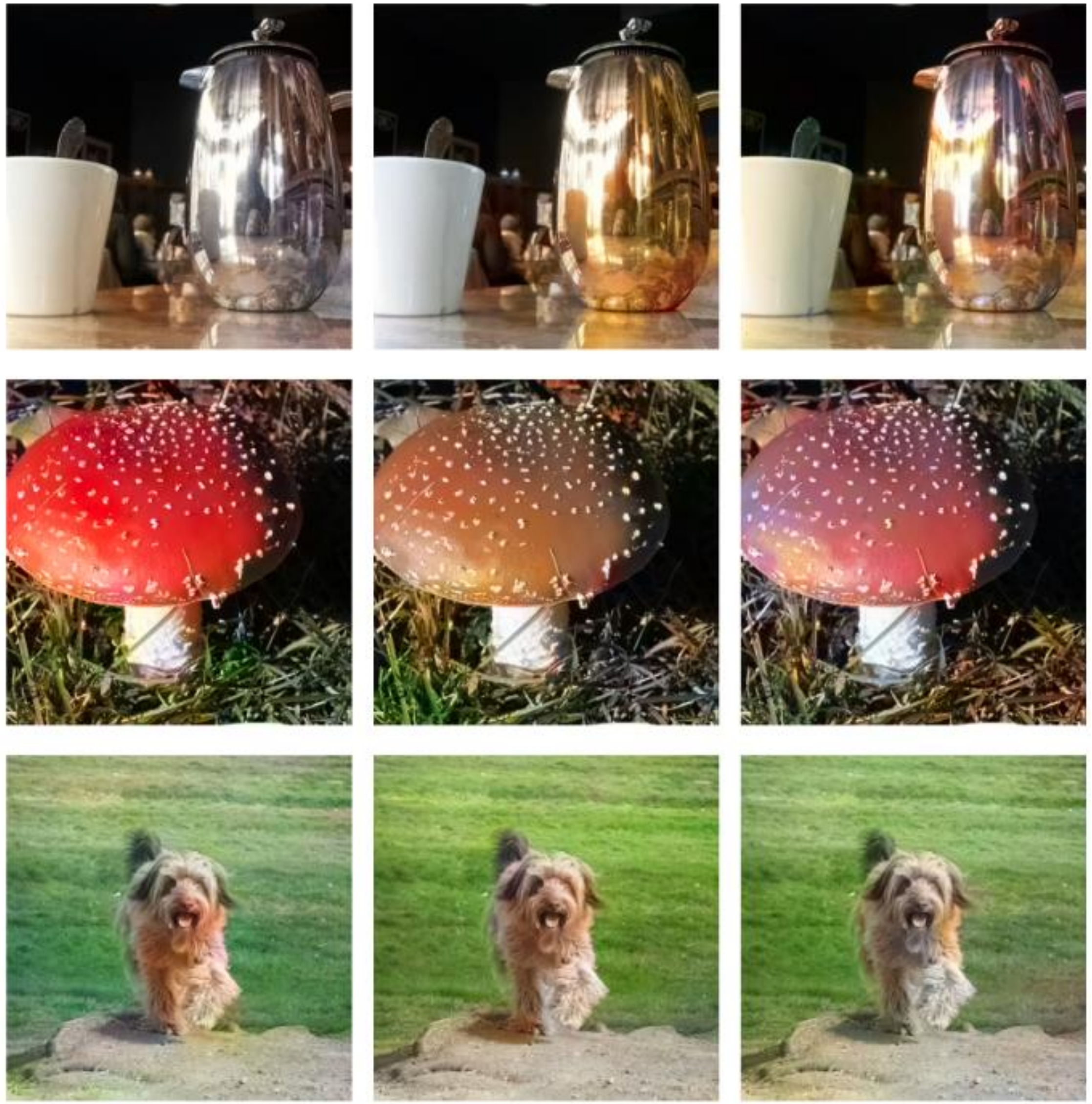
- Multi-scale decomposition via *Haar-Wavelet* down-sampling (standard max pooling not invertible)



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

Colorization Examples

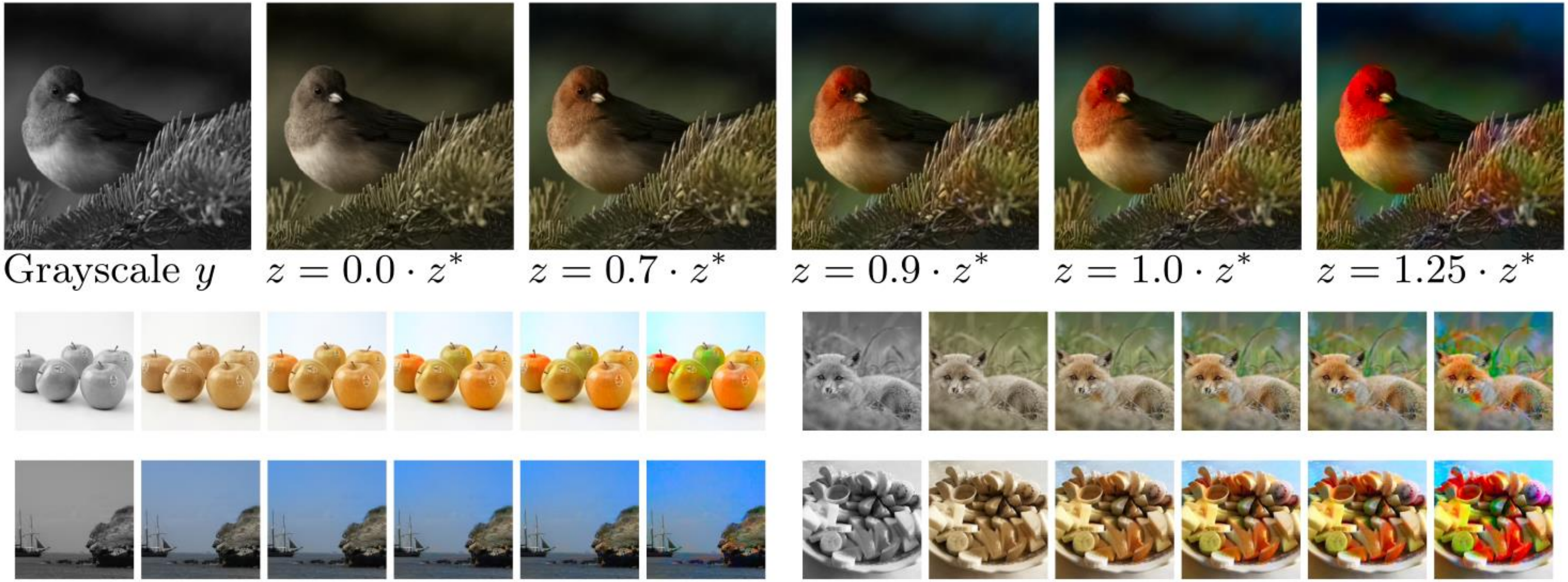




Ardizzone et al. **"Guided Image Generation with Conditional Invertible Neural Networks"**, arXiv 2019.

Colorization: Meaningful Latent Manipulations

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



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

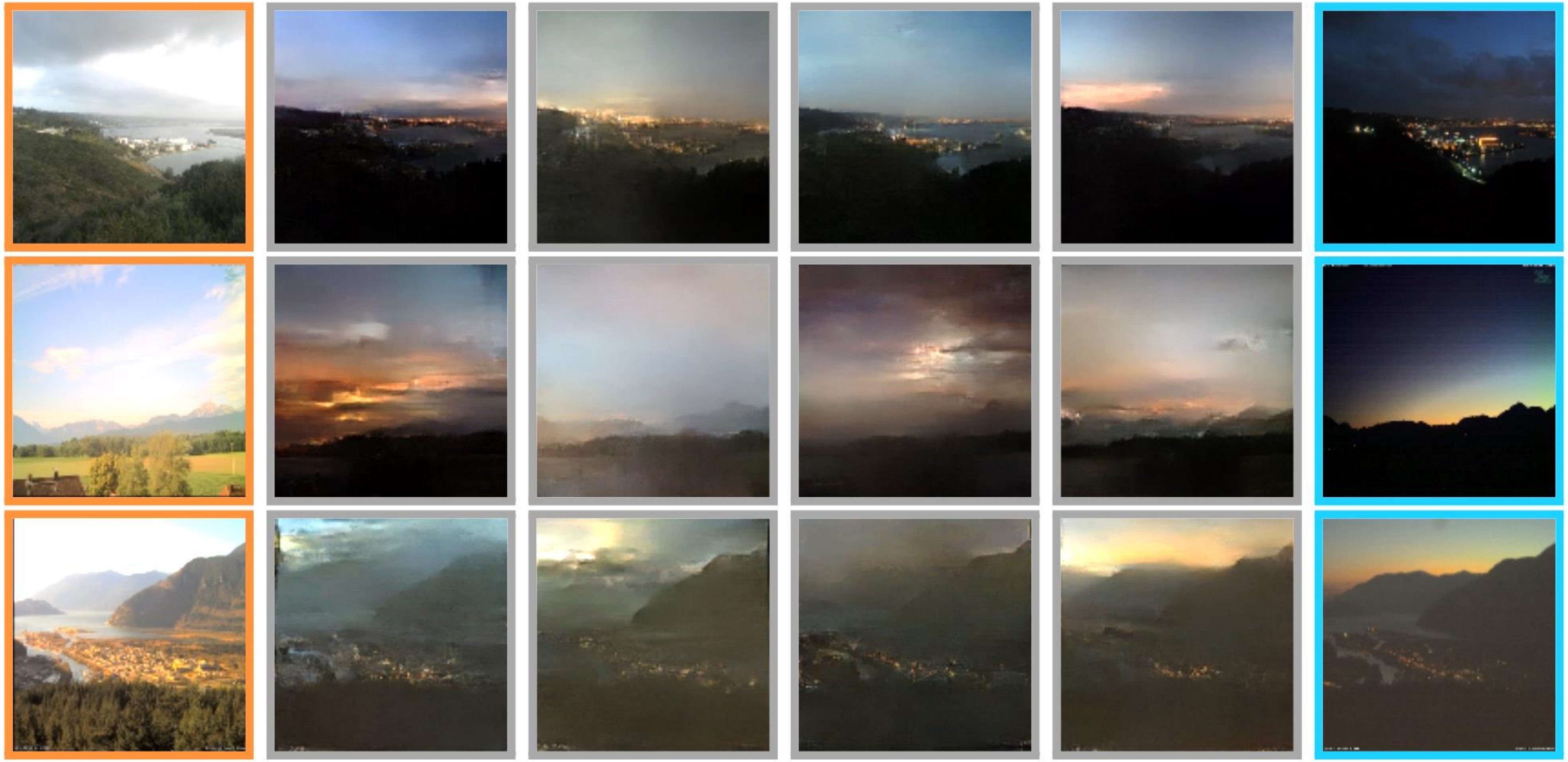
Colorization: Meaningful Latent Manipulations

- Color transfer
  - Encode color of input image $i = [L_i, a_i, b_i]$: $z_i = f(x = [a_i, b_i]; h'(y = L_i))$
  - Reconstruct color for a different grayscale image L_c: $\hat{x}_i = [\widehat{a_i}, \widehat{b_i}] = g(z_i;\ h'(y = L_c))$ with $g = f^{-1}$ while *keeping* the latent code z_i

cINN for Image-to-Image Transformation

- 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 Transformation

- Results:

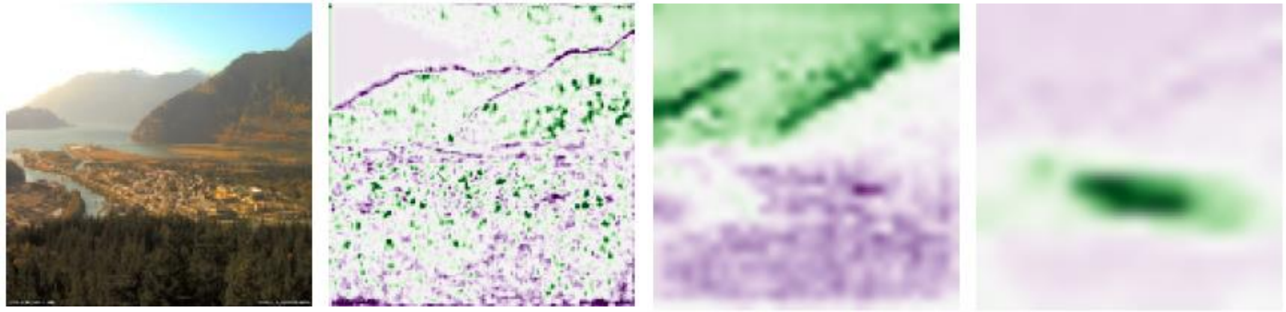
Condition *y*
Day image

Generated *x*
Night images

Ground truth *x*
Night image

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

Day image
Level 1
Level 2
Level 3



Solving Inverse Problems with Invertible Neural Networks

Ullrich Köthe
Visual Learning Lab, Heidelberg University
joint work with **Lynton Ardizzone, Stefan Radev, Jakob Kruse, Tim Adler, Carsten Rother, Lena Maier-Hein**

Tutorial „Normalizing Flows“ at CVPR 2021
June 2021

Towards an INN-based solution: 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 $z = x_1 - x_2$ ⇨ $(y, z) = g_{\text{aug}}(x_1, x_2) = (x_1 + x_2, x_1 - x_2)$ is invertible!
  - Reparametrize posterior $p(x_1, x_2 \mid y)$ as $(x_1, x_2) = g_{\text{aug}}^{-1}(y, z) = \left(\frac{y+z^{(t)}}{2}, \frac{y-z^{(t)}}{2}\right)$ with $z \sim \mathcal{N}(0,2)$
    - Given actual observation $\hat{y}$, repeat 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 $p(x_1, x_2 \mid \hat{y})$

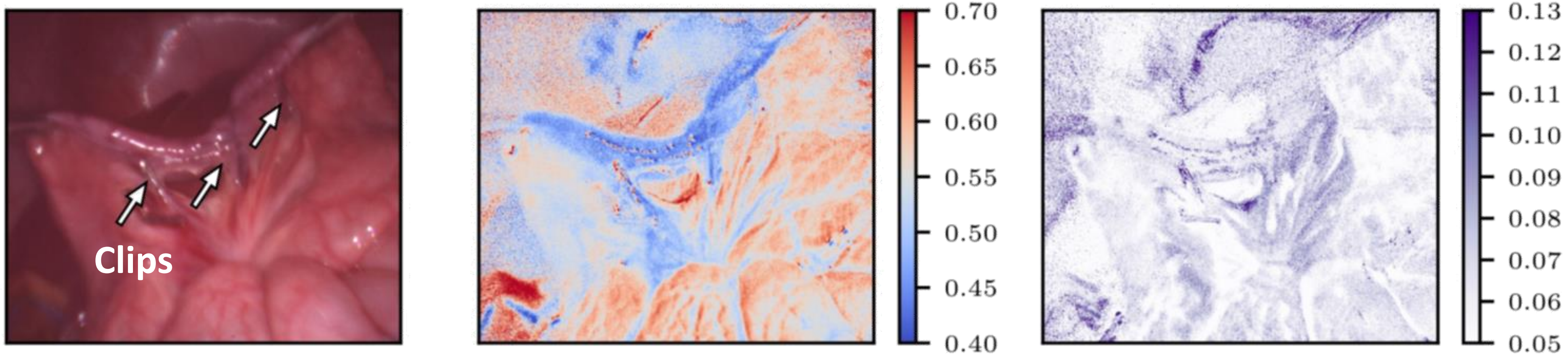
Generalize this to complex settings (non-linear g, noise, high dimensions) by INNs.

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 color spectrum y

- **Task:** given spectrum, find posterior distribution of tissue state parameters
- Forward process $s(x)$ is implemented by Monte Carlo simulation



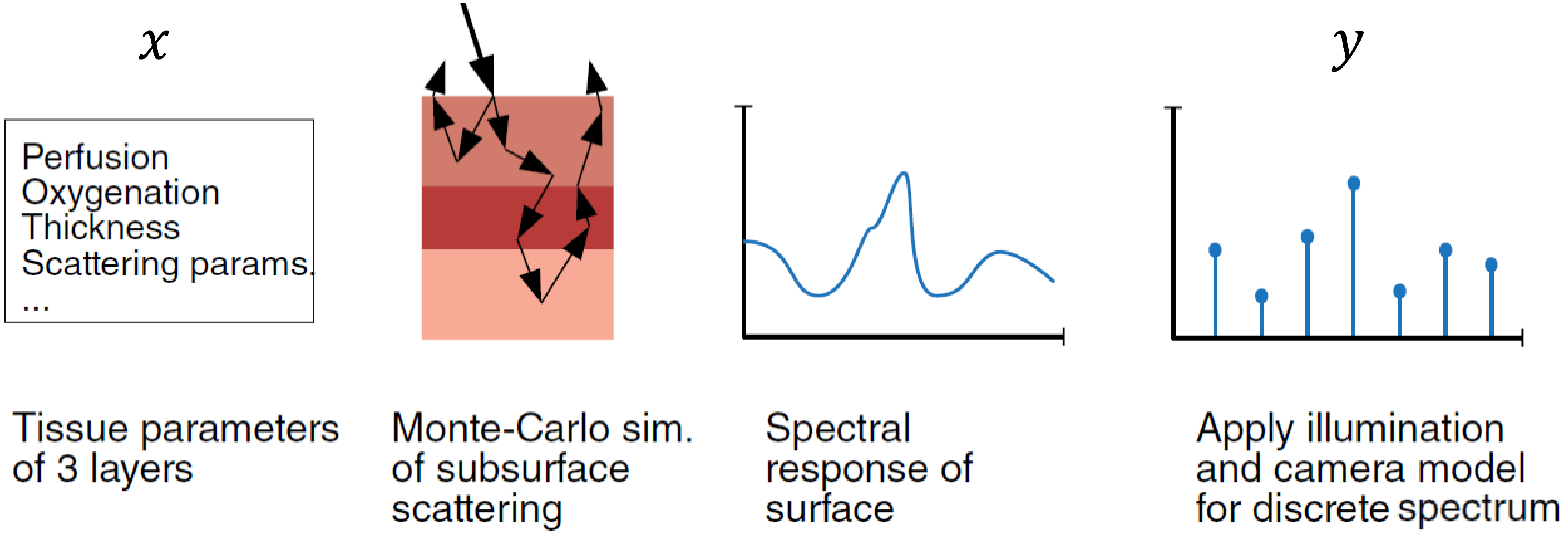
c) RGB image a) Median s_{O_2} b) Est. uncertainty

Ardizzone et al. **“Analyzing inverse problems with invertible neural networks”**, ICLR 2019.

Application: Multispectral Endoscopy

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

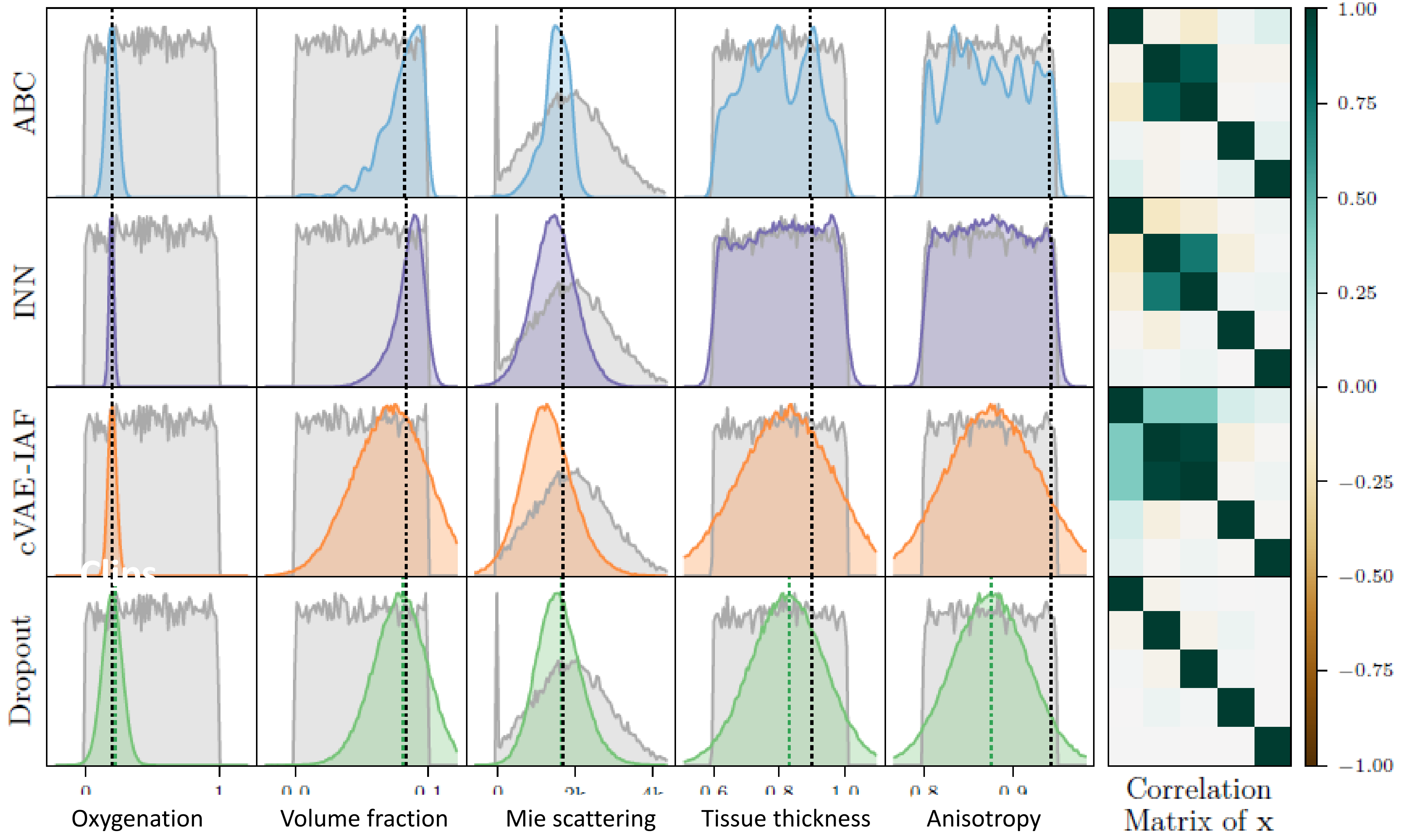
- training: INN learns $[y, z] = f_\theta(x) \approx s_{aug}(x)$ with $p(z) \sim \mathcal{N}(0, \mathbb{I})$
- inference: given observed spectrum $\hat{y}$, sample $\{z_i \sim p(z)\}_{i=1}^M$ and compute posterior sample $\left\{x_i = f_\theta^{-1}(\hat{y}, z_i)\right\}_{i=1}^M$ (independently for every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



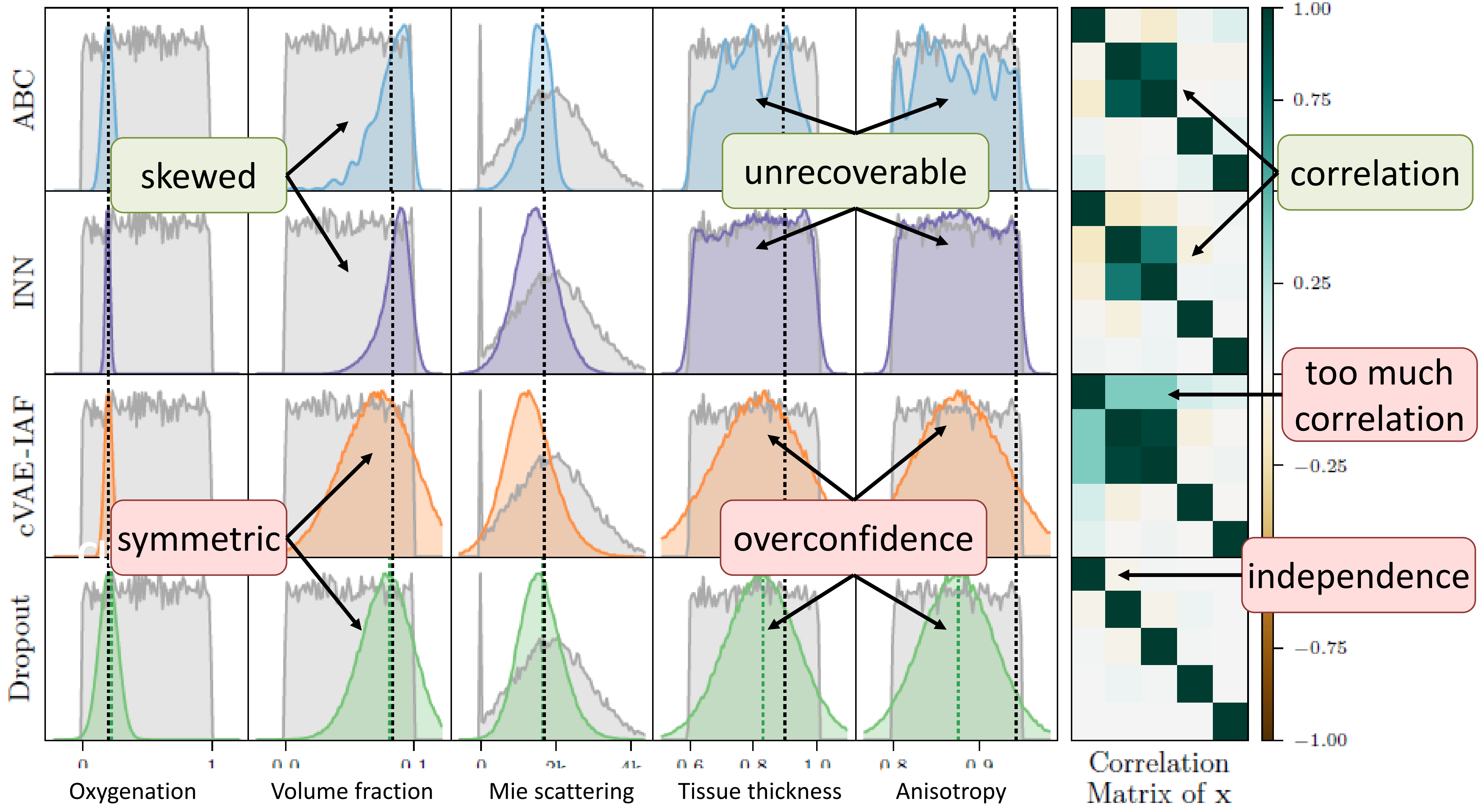
Ardizzone et al. **“Analyzing inverse problems with invertible neural networks”**, ICLR 2019.

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

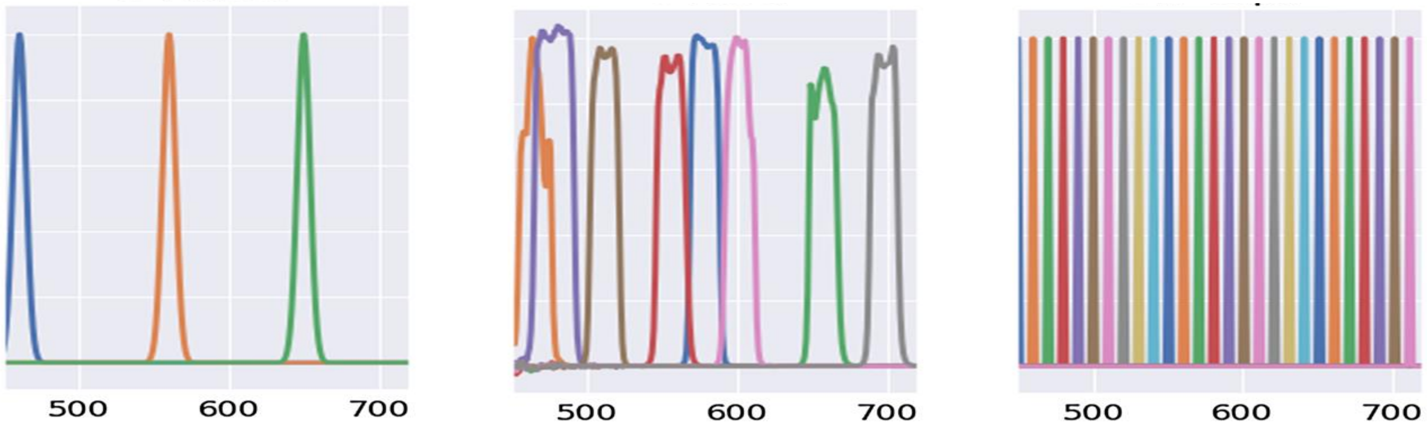


Experimental Design for Multispectral Endoscopy

Analysis of posteriors: Which camera should be used?

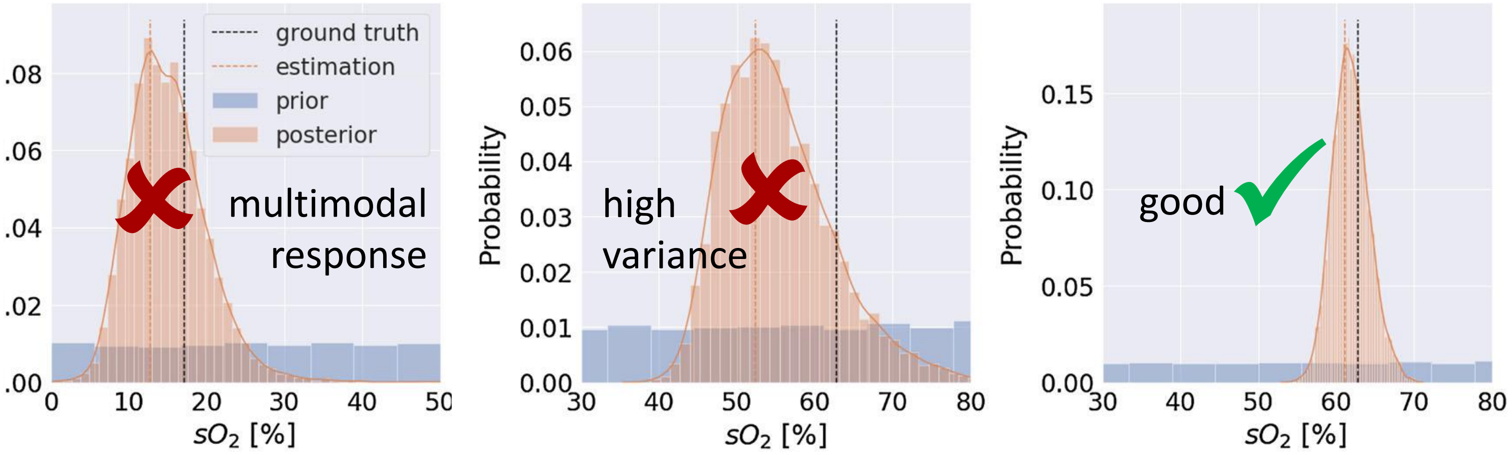
- 3 to 27 spectral channels
- Which gives reliable results at best price and usability?

3 spectral channels | 8 spectral channels | 27 spectral channels



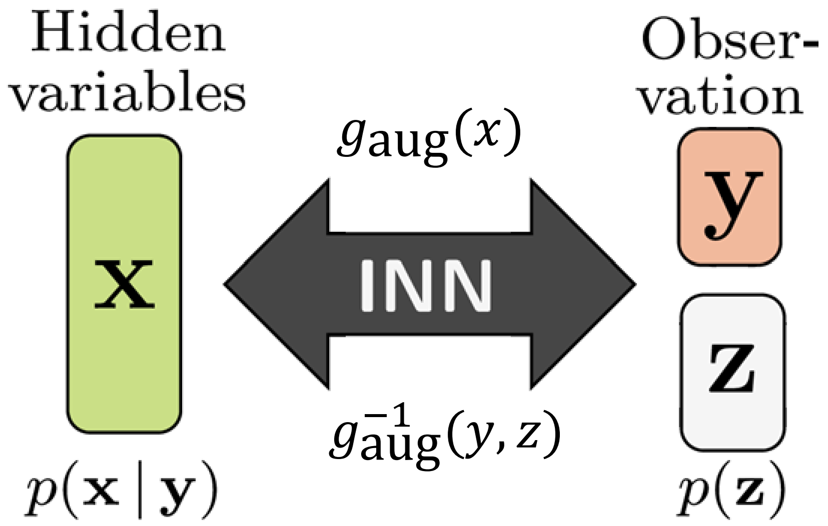
- posterior oxygen level histograms:

⇨ 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 Conditional Inference

Split latent space

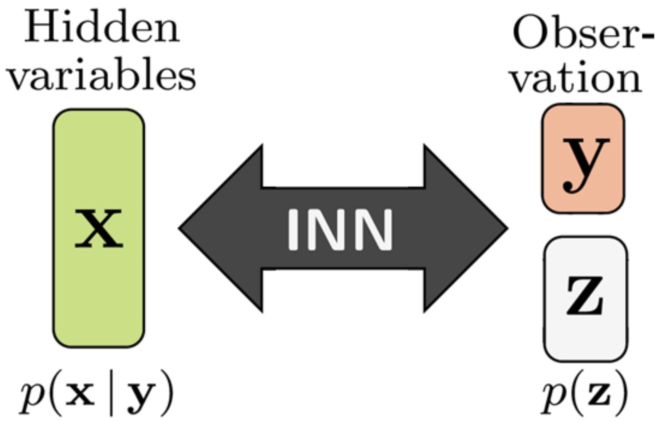
training: $(y, z) = f_\theta(x)$

s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$

compute $x = f_\theta^{-1}(\hat{y}, z)$

⇨ $x \sim p(x \mid \hat{y})$



historically first

Latent mixture INN

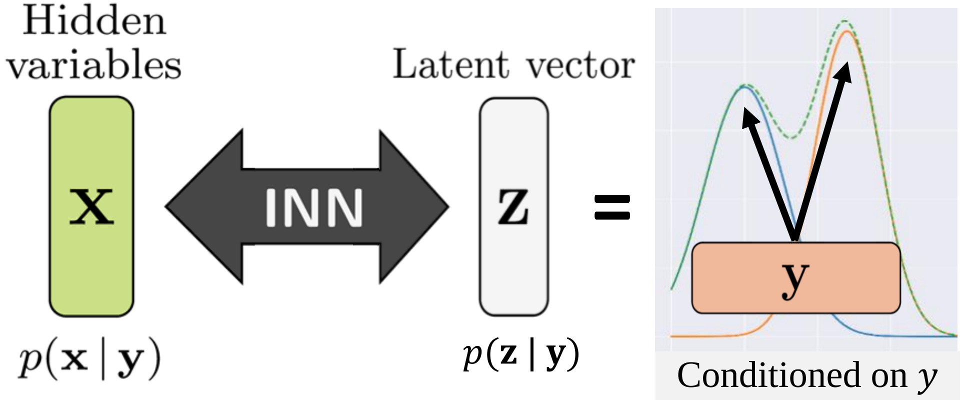
training: $z = f_\theta(x)$

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 $x = f_\theta^{-1}(z)$

⇨ $x \sim p(x \mid \hat{y})$



classification, disentanglement

Conditional INN

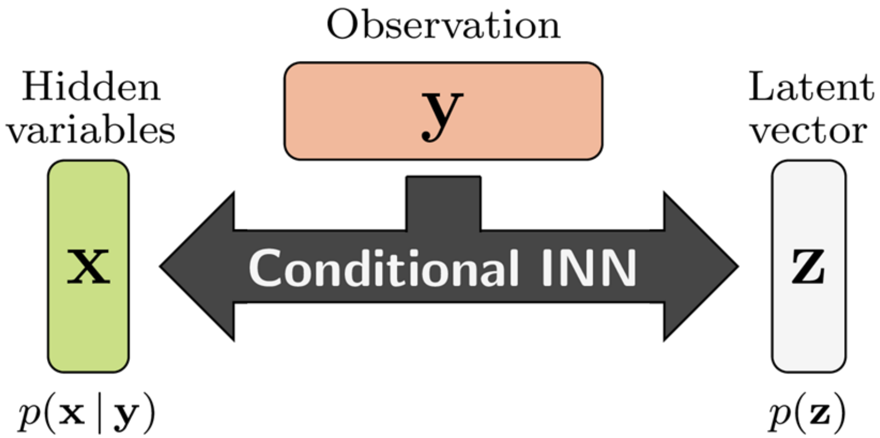
training: $z = f_\theta(x; y)$

s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$

compute $x = f_\theta^{-1}(z; \hat{y})$

⇨ $x \sim p(x \mid \hat{y})$



inverse inference

BayesFlow: Model-Based Inverse Inference with cINNs

Model-based inverse inference:

- system with intrinsic parameters x (hidden) and observations y (measurable)
- good scientific understanding of the forward process: How does y arise from given x? (e.g. differential equations, simulations)
- solve the inverse problem: Which hidden parameters x explain some actual observations $\hat{y}$?
- usually no analytic solution
  - ambiguous outcomes due to information loss from x to y ⇨ must estimate posterior $p(x \mid \hat{y})$
  - simplest approach: manually adjust x until outcomes match $\hat{y}$ ⇨ neglects uncertainty
  - traditional Bayesian inference: sampling methods (MCMC, HMC, …) ⇨ very expensive
- standard ML methods are often not applicable
  - lack of training data with known ground truth x^*
  - only point estimates, no posteriors (i.e. no diverse outputs)
- cINN can elegantly solve the Bayesian inverse problem

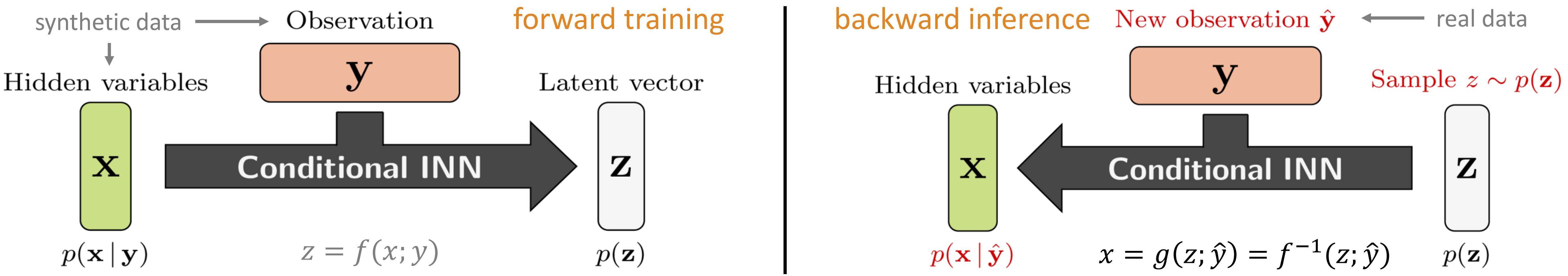
BayesFlow: Model-Based Inverse Inference with cINNs

cINNs make clever use of the known forward model to solve the inverse problem

- run cINN in forward mode for model-based training
  - use known forward model to create synthetic training data ⇨ cINN becomes a fast surrogate
  - train with diverse forward scenarios and noise ⇨ cINN learns the ambiguity and uncertainty
- run cINN in backward mode for inverse inference
  - use actual observations $\hat{y}$ as condition
    - sample many latents $\{ z_k \sim p(z) \}_{k=1}^N$
    - run cINN backwards $\{ x_k = g(z_k; \hat{y}) \}_{k=1}^N$

$\Big\}$ the $\{x_k\}$ are a sample of the Bayesian posterior $p(x \mid \hat{y})$

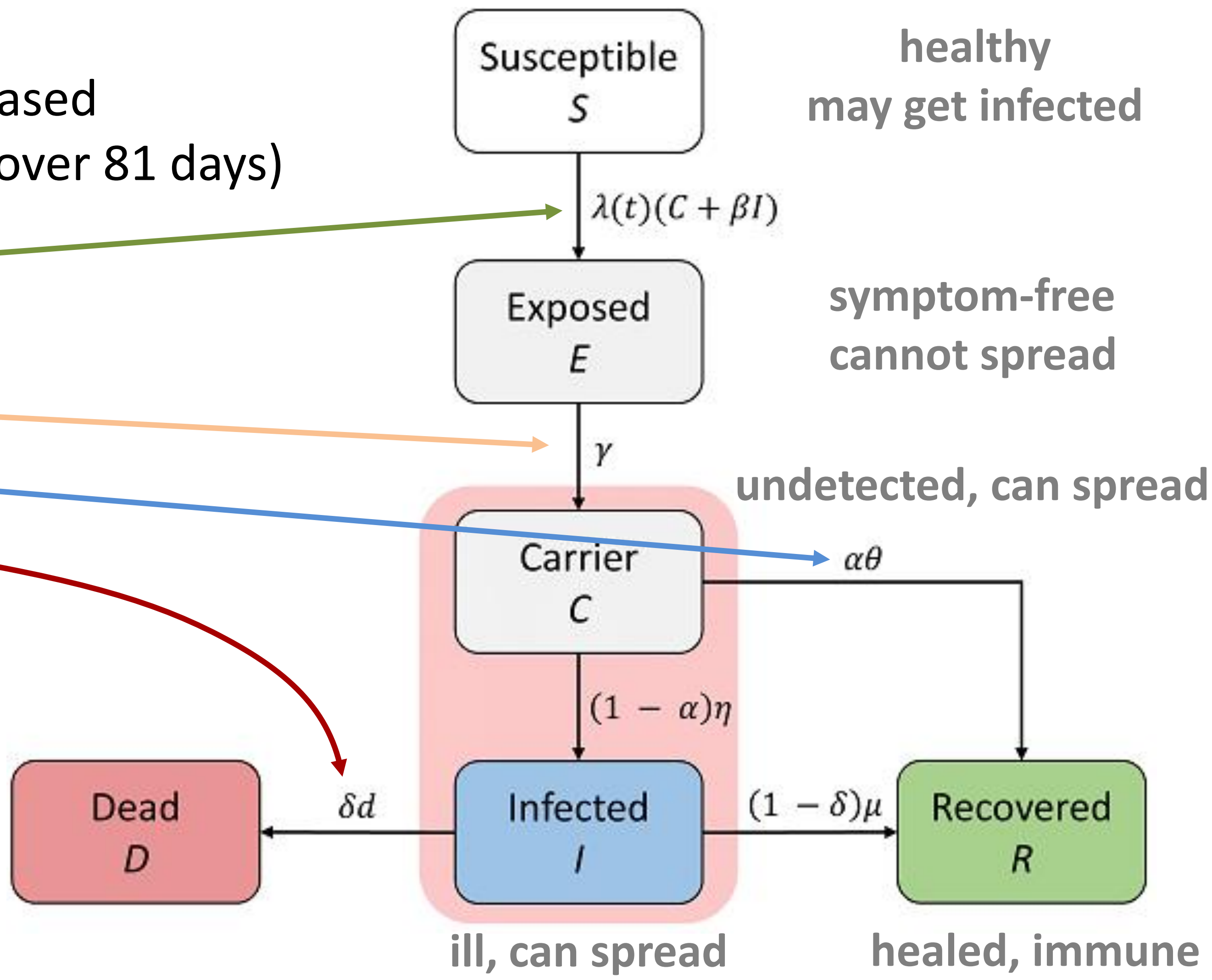
"Train forward, get the inverse for free."



BayesFlow for Covid-19 Epidemiology

Epidemiology as a difficult inverse problem:

- observations: time series of infected, recovered, deceased
  (as of June 2020: 243 measurements = 3 observables over 81 days)
- 34 hidden parameters:
  - infection rate $\lambda(t)$
  - latent period $1/\gamma$
  - undetected fraction α
  - case fatality rate δ
  - …
- prior knowledge:
  - SIR-type compartmental model
    (ODE system similar to Lotka-Voterra)
  - dates of government interventions
  - sources of reporting errors
  - …



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

Forward Model: Epidemic Calculator

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

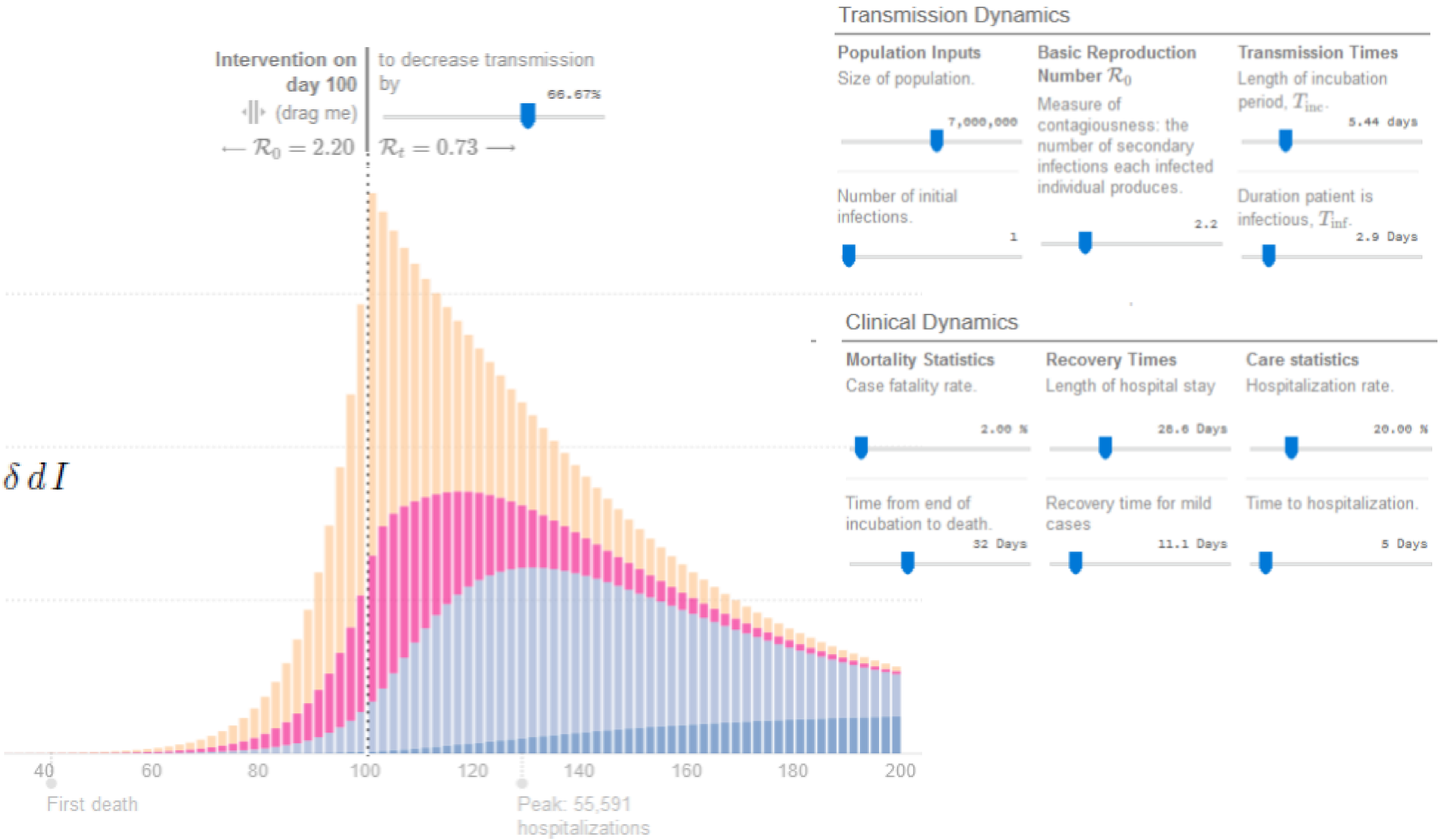
$$\frac{dE}{dt} = \lambda(t)\left(\frac{C+\beta I}{N}\right) S - \gamma E$$

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

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

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

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



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

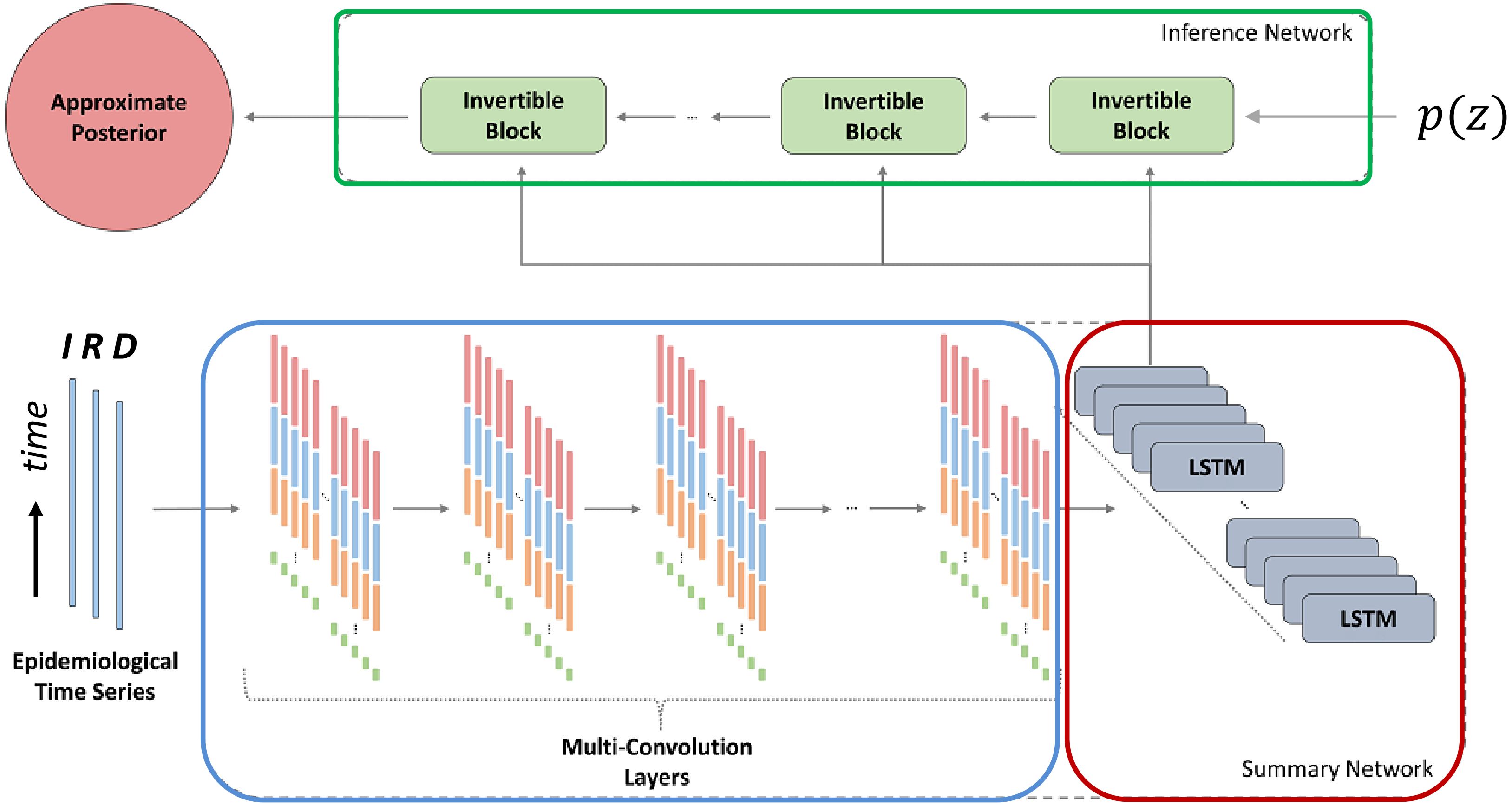
BayesFlow for Epidemiology: The Networks

- Inference problem: observation sequence (IRD) ⇨ parameter posteriors
  - Solve with BayesFlow network: cINN with statistical preprocessing networks for y
  - Training: end-to-end optimization of maximum likelihood loss with 70000 simulations

Convolutional:
- noise reduction
  feature detection

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

Invertible (cINN):
- posterior inference

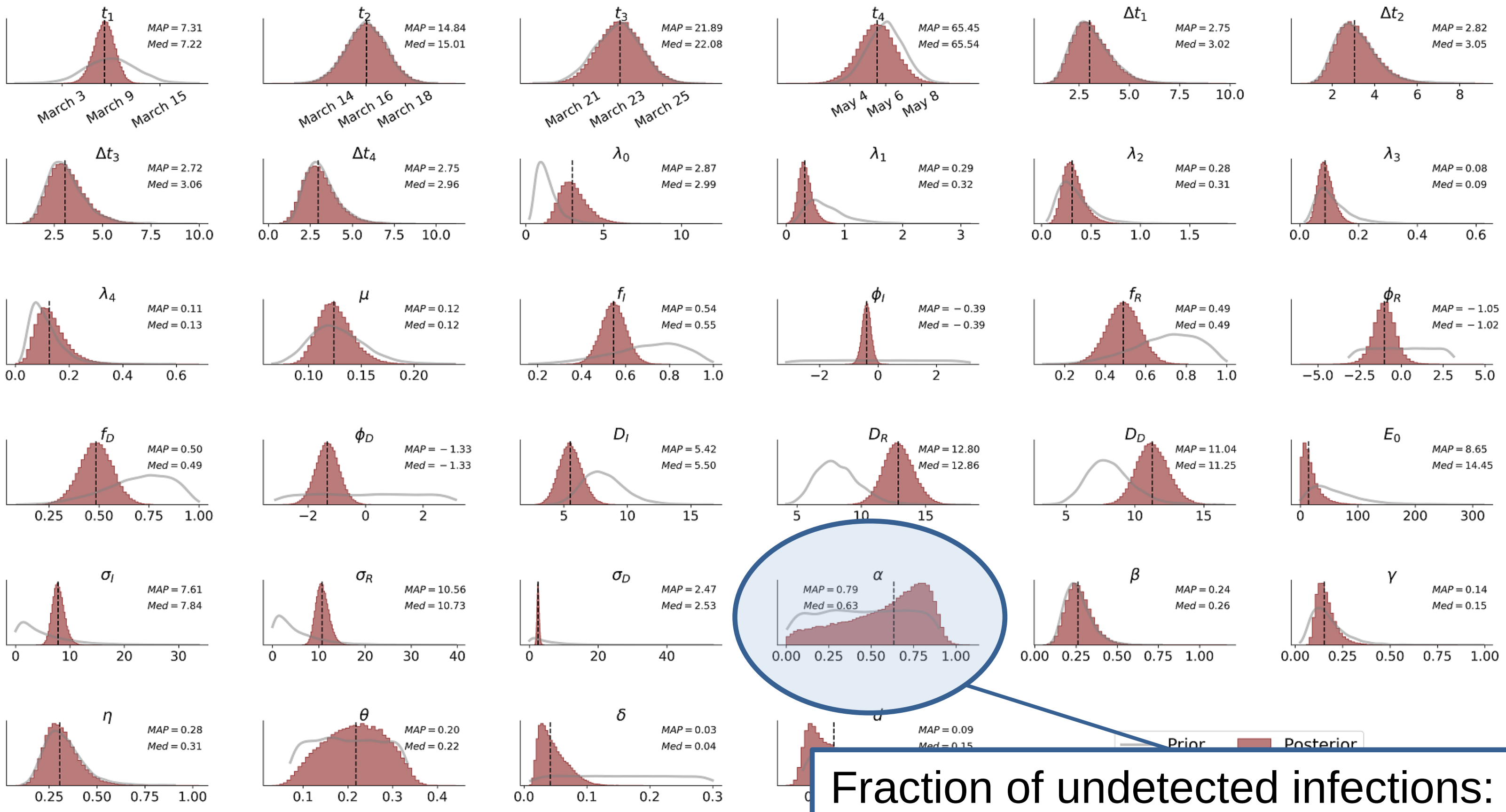


Radev et al. **"Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany"**, arXiv 2020.

BayesFlow for Epidemiology: Covid-19 Marginal Posteriors

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

- High fraction of undetected infections: 63% (median), 79% (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



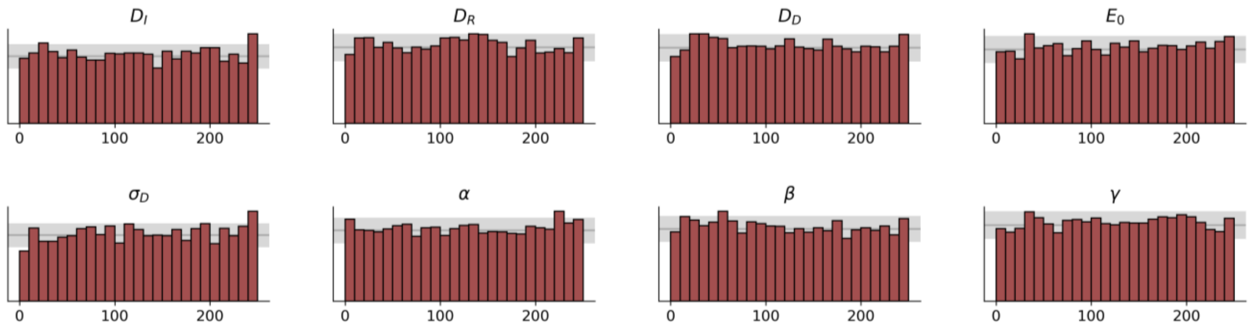
Fraction of undetected infections:
uniform prior ⇨ peaked posterior

Correspond well to clinical findings

BayesFlow for Epidemiology: Strengths

Well-calibrated uncertainty quantification

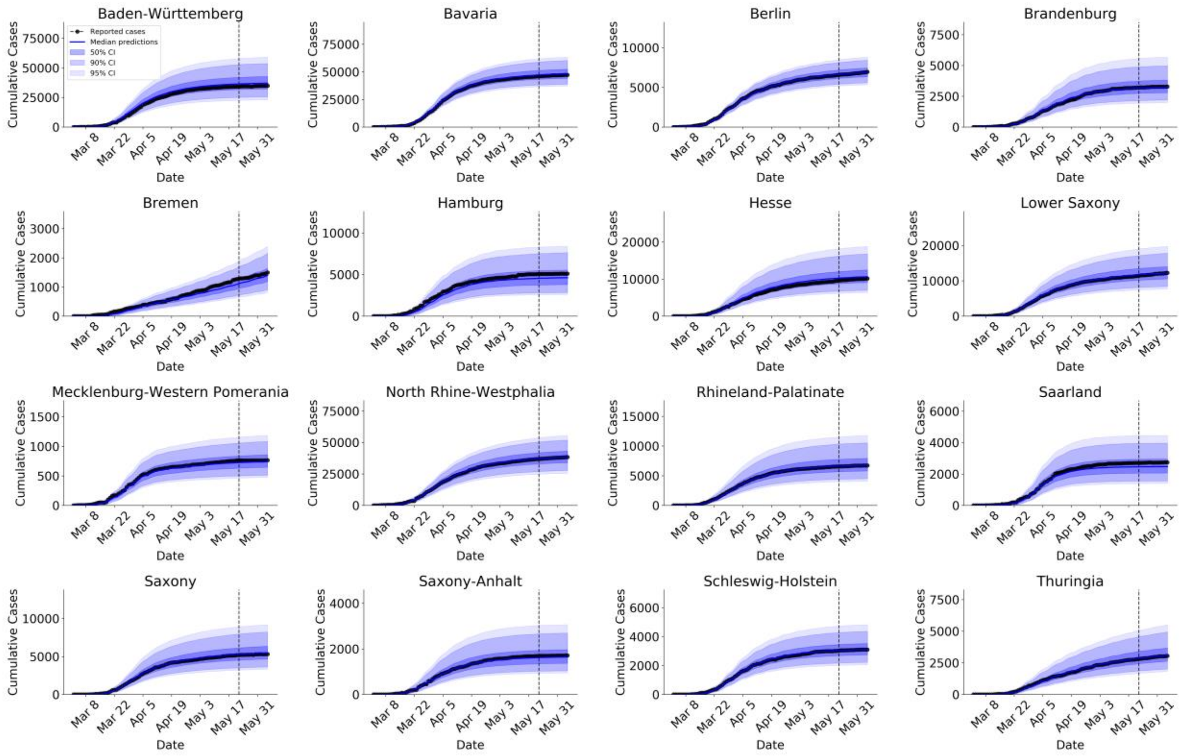
- $q\%$ confidence intervals are hit $\approx q\%$ of the time
- much better than classical estimators (e.g. least squares fitting, manual parameter tuning, …)



Efficient backward operation ⇨ fast inference

- train once, predict often
- in contrast, MCMC runs from scratch for each $\hat{y}$
- Bayesflow upfront training effort $\approx 10 - 100$x of single MCMC inference

⇨ training effort amortizes quickly



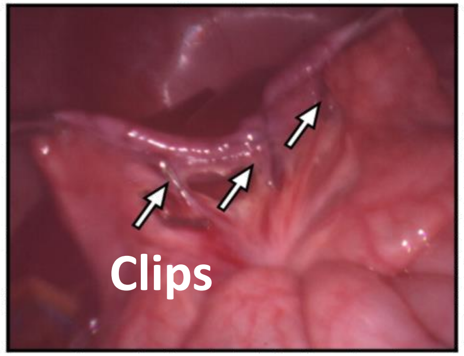
analysis of German states with identical network

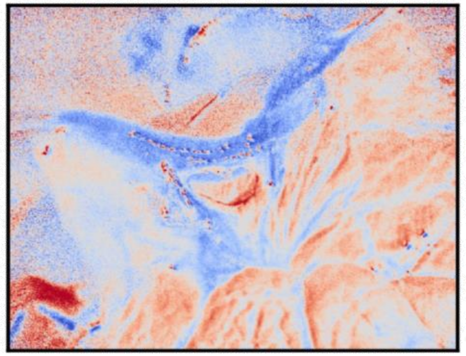
Radev et al. **"Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany"**, arXiv 2020.

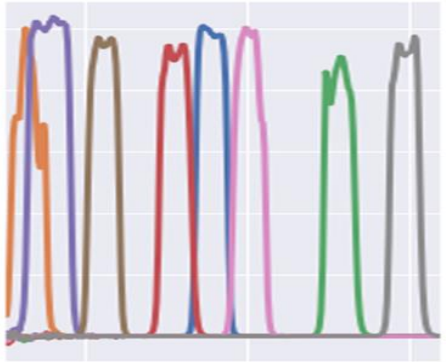
Very diverse inverse problems were solved with INNs/BayesFlow

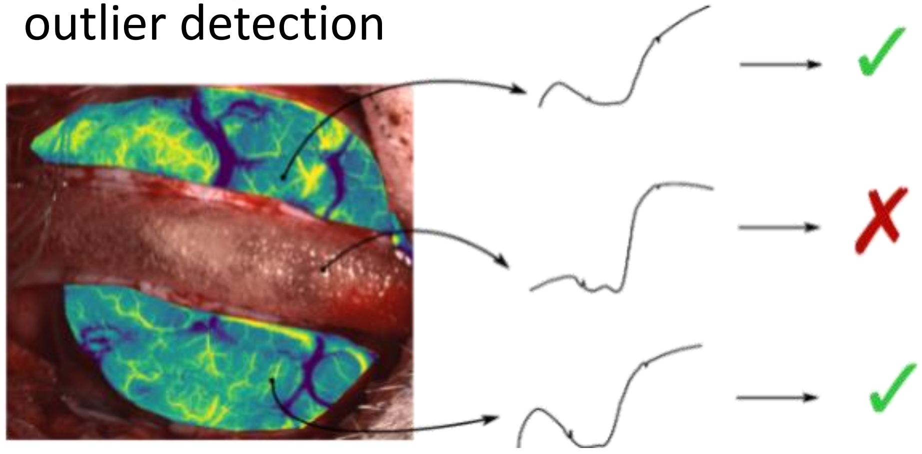
- Surgery:

blood oxygenation | experimental design | outlier detection



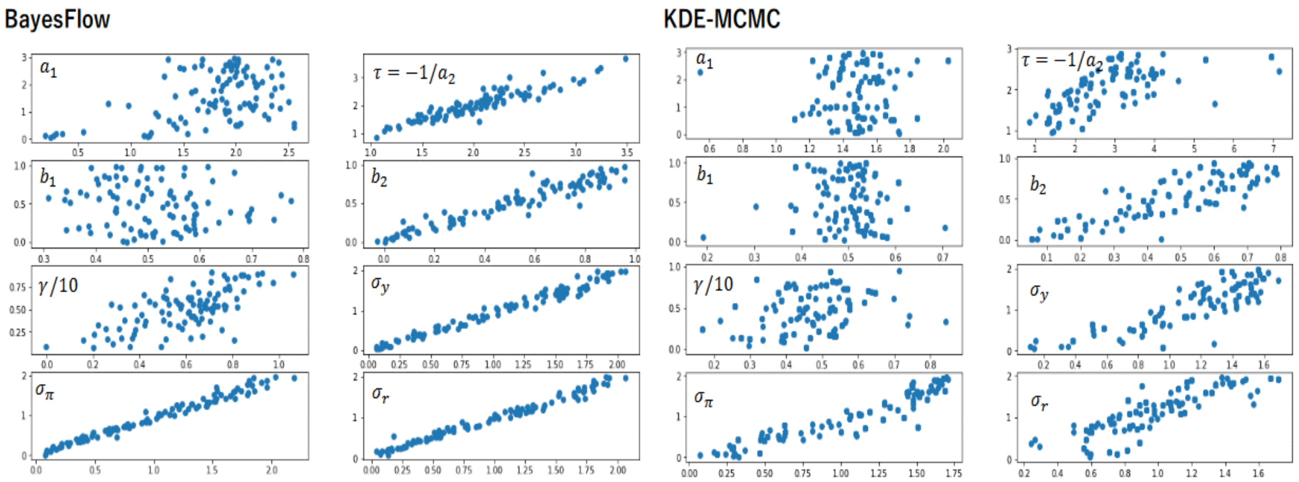






- Photo-acoustic imaging
- Particle physics
- Astrophysics
- Environmental physics
- Cognitive Science
- Inverse kinematics of robots
- Mechanical engineering

- Finance:



BayesFlow beets MCMC

Ardizzone et al. **"Analyzing inverse problems with invertible neural networks"**, ICLR 2019.
Adler et al. **"Out of distribution detection for intra-operative functional imaging"**, UNSURE 2019.
Shiono **"Estimation of agent-based models using BayesFlow"**, SSRN 2020.

Guaranteed disentanglement with Nonlinear ICA and Incompressible Flows

Ullrich Köthe
Visual Learning Lab, Heidelberg University
joint work with **Carsten Rother, Peter Sorrenson**

Tutorial „Normalizing Flows“ at CVPR 2021
June 2021

INN Architectures for Conditional Inference

Split latent space

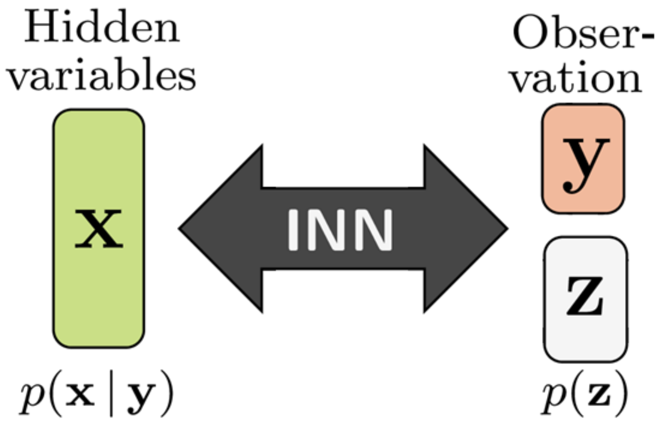
training: $(y, z) = f_\theta(x)$

s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$

compute $x = f_\theta^{-1}(\hat{y}, z)$

⇨ $x \sim p(x \mid \hat{y})$



historically first

Latent mixture INN

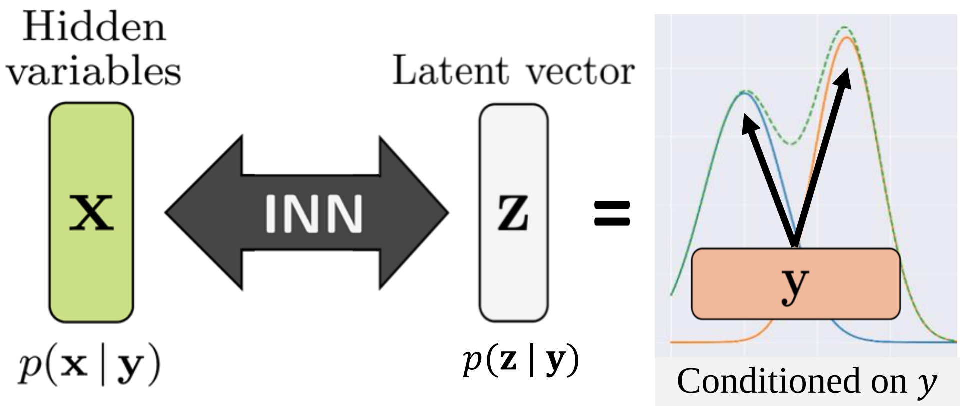
training: $z = f_\theta(x)$

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 $x = f_\theta^{-1}(z)$

⇨ $x \sim p(x \mid \hat{y})$



classification, disentanglement

Conditional INN

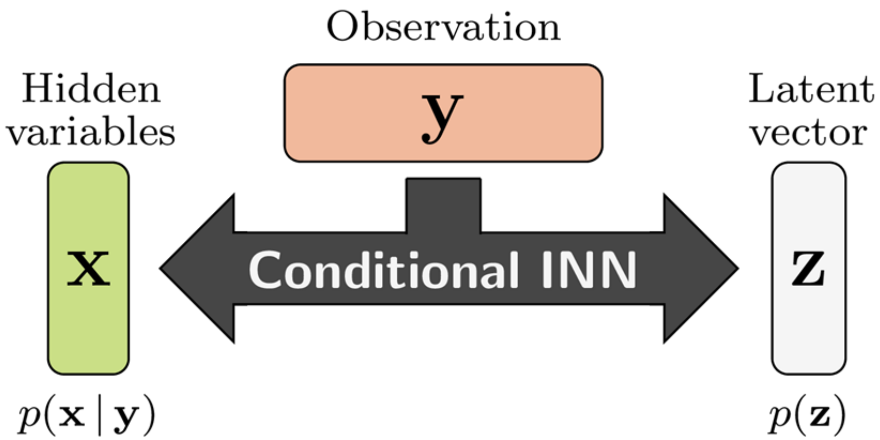
training: $z = f_\theta(x; y)$

s.t. $p(z) = \mathcal{N}(0, \mathbb{I})$

inference: sample $z \sim \mathcal{N}(0, \mathbb{I})$

compute $x = f_\theta^{-1}(z; \hat{y})$

⇨ $x \sim p(x \mid \hat{y})$

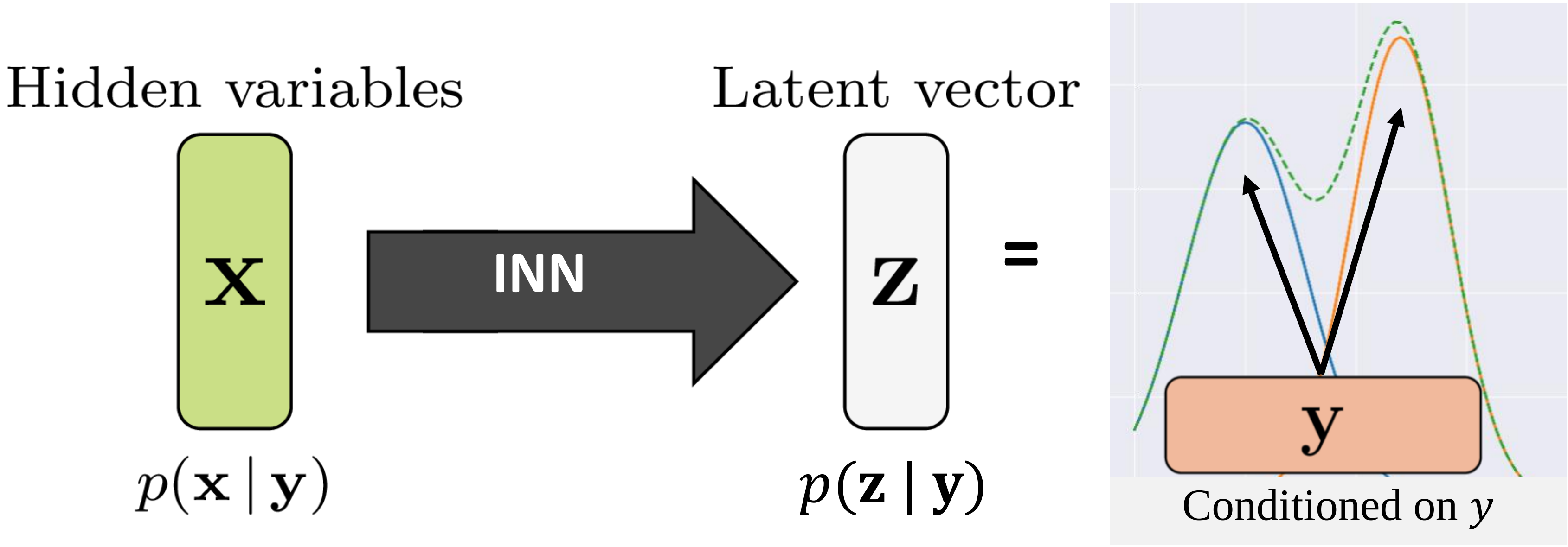


inverse inference

Interpretable Latent Spaces with Latent Mixture INNs (LM-INNs)

Interpretable latent spaces are a key to explainable machine learning

- Latent Mixture INNs are especially suitable for this task
  - Variation of cINNs: condition y acts on the latent space, not on the function g
    - cINN: $x \sim p(x \mid y) \iff z \sim p(z), \quad x = g(z; y)$
    - LM-INN: $x \sim p(x \mid y) \iff z \sim p(z \mid y), \quad x = g(z)$
  - Define $p(z \mid y)$ as a mixture of Gaussians instead of a single Gaussian
  - Especially simple when y is a class label: learn one mixture component $p(z \mid y = k)$ per label k

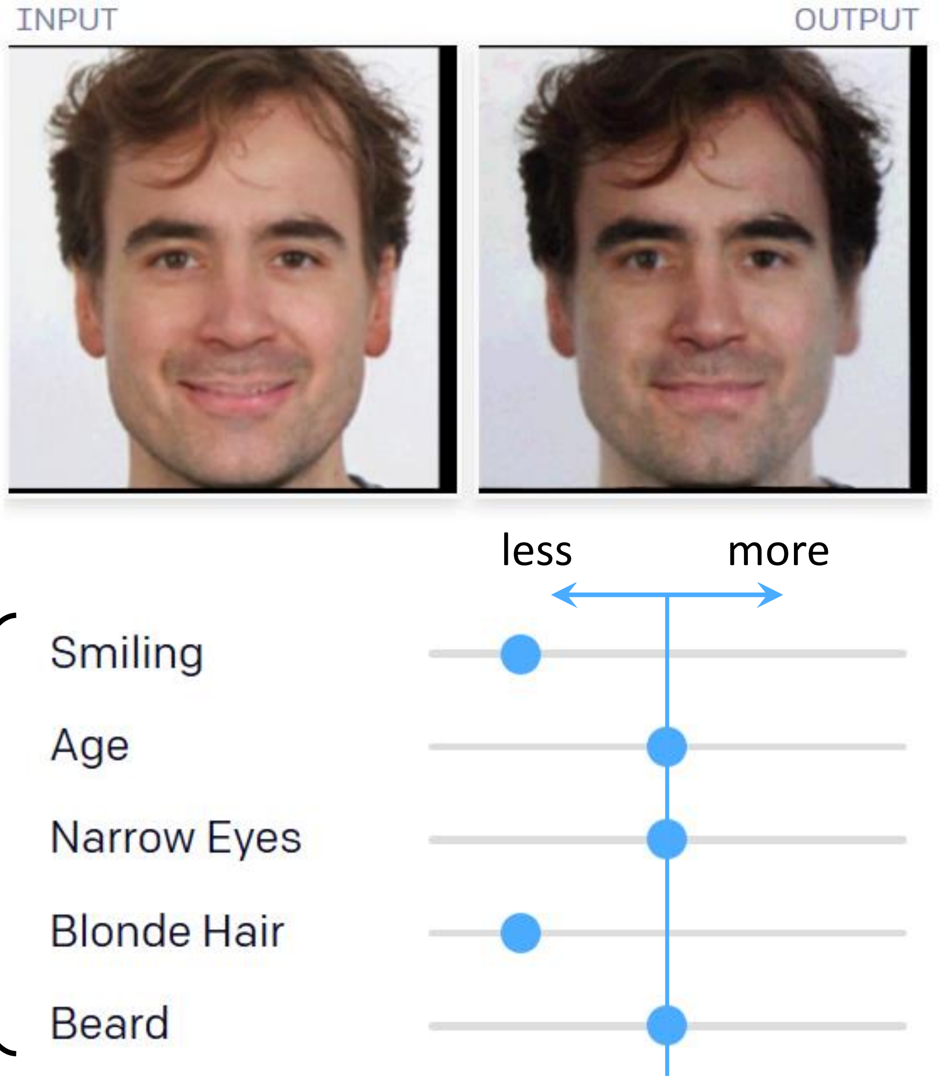


What is Disentanglement?

Train network so that each latent feature has a single interpretable effect

Example: GLOW INN

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



changing the value of a single latent feature has a coordinated and intuitive effect on many pixels simultaneously

Kingma & Dhariwal **"Glow: Generative flow with invertible 1x1 convolutions"**, NIPS 2018.

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.

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.

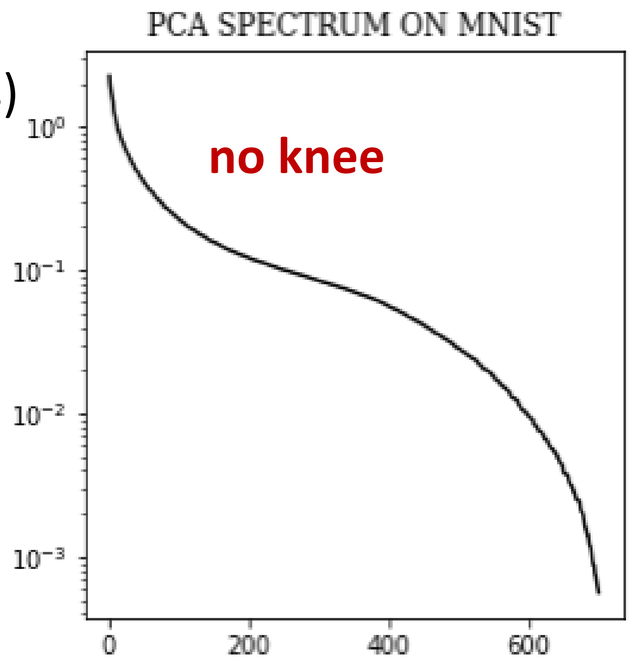
LM-INNs for Disentanglement

- Fundamental disentanglement: separate content from noise
  - number of content dimensions = intrinsic dimension of the dataset
  - similar to autoencoder bottleneck, but intrinsic dimension is learned (not chosen as a hyperparameter)
- Content disentanglement: disentangle content subspace into meaningful features

Classical: linear disentanglement by PCA

- do eigen decomposition
- sort features (eigenvectors) by energy (eigenvalues)

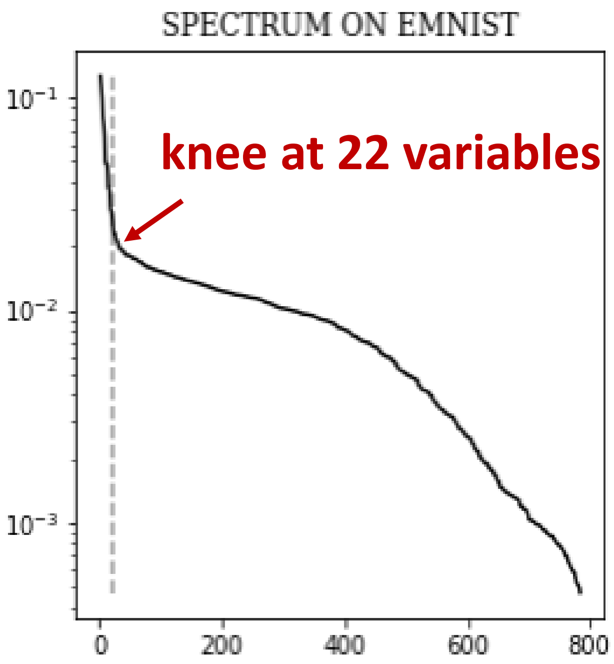
Spectrum is usually smooth
⇨ no clear choice for intrinsic dimension



New: non-linear ICA by LM-INN

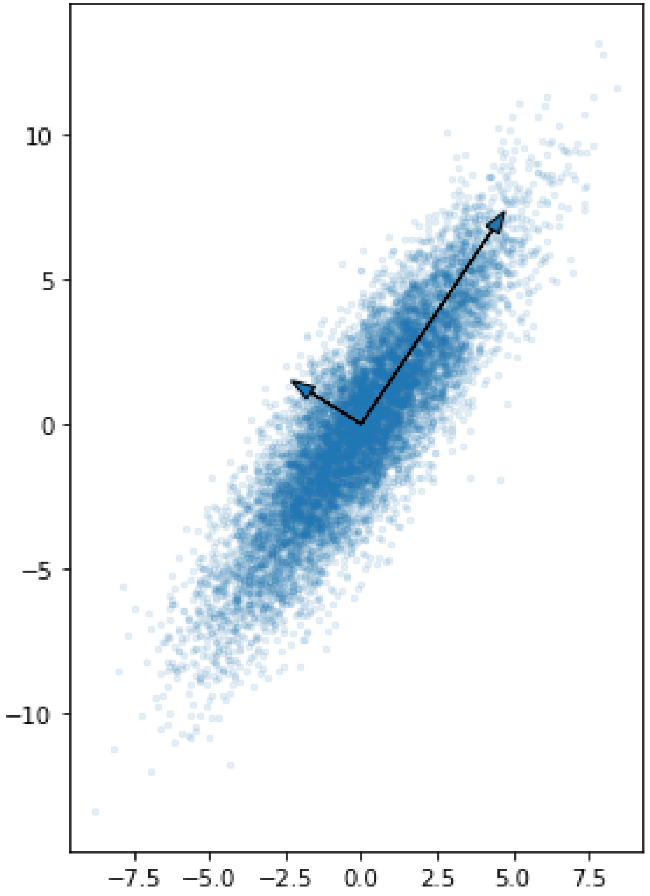
- train LM-INN (y = class labels)
- sort features (latent variables) by energy (latent variance)

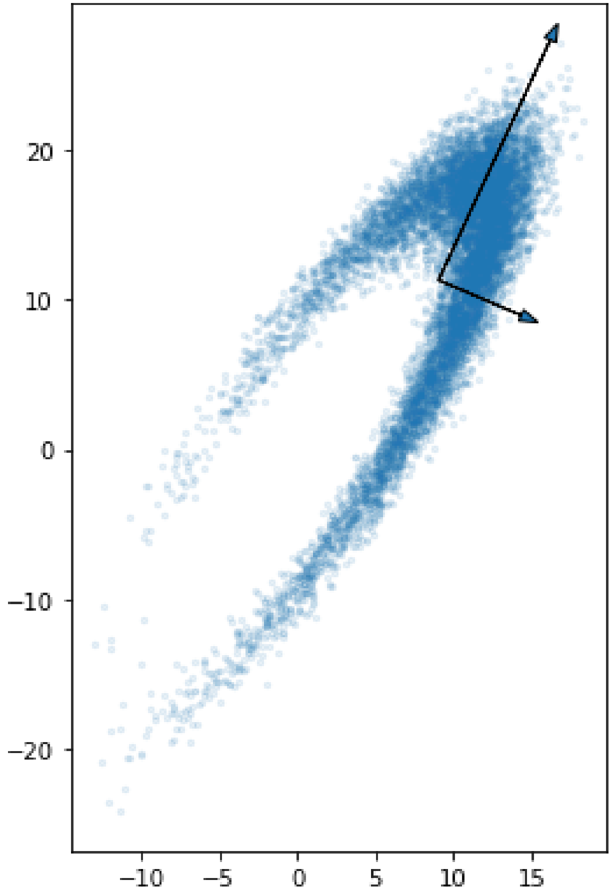
Spectrum may have marked knee
⇨ clear identification of the intrinsic dimension



Recap: PCA (Principal Component Analysis)

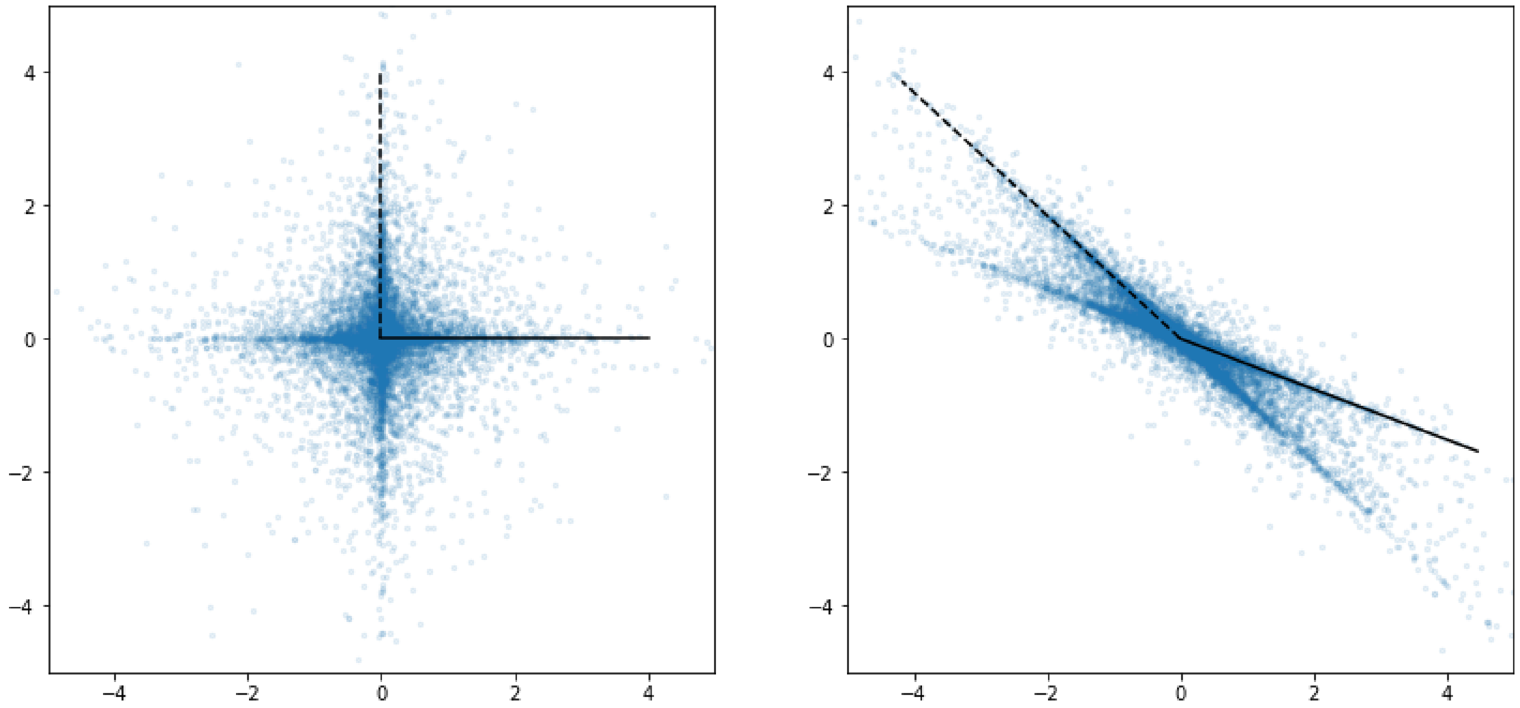
- Classical method for unsupervised disentanglement with a linear transformation
  - Finds uncorrelated basis vectors for multivariate Gaussian distributions
  - Can be applied to non-Gaussian data, but cannot fully disentangle them





Recap: ICA (Independent Component Analysis)

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



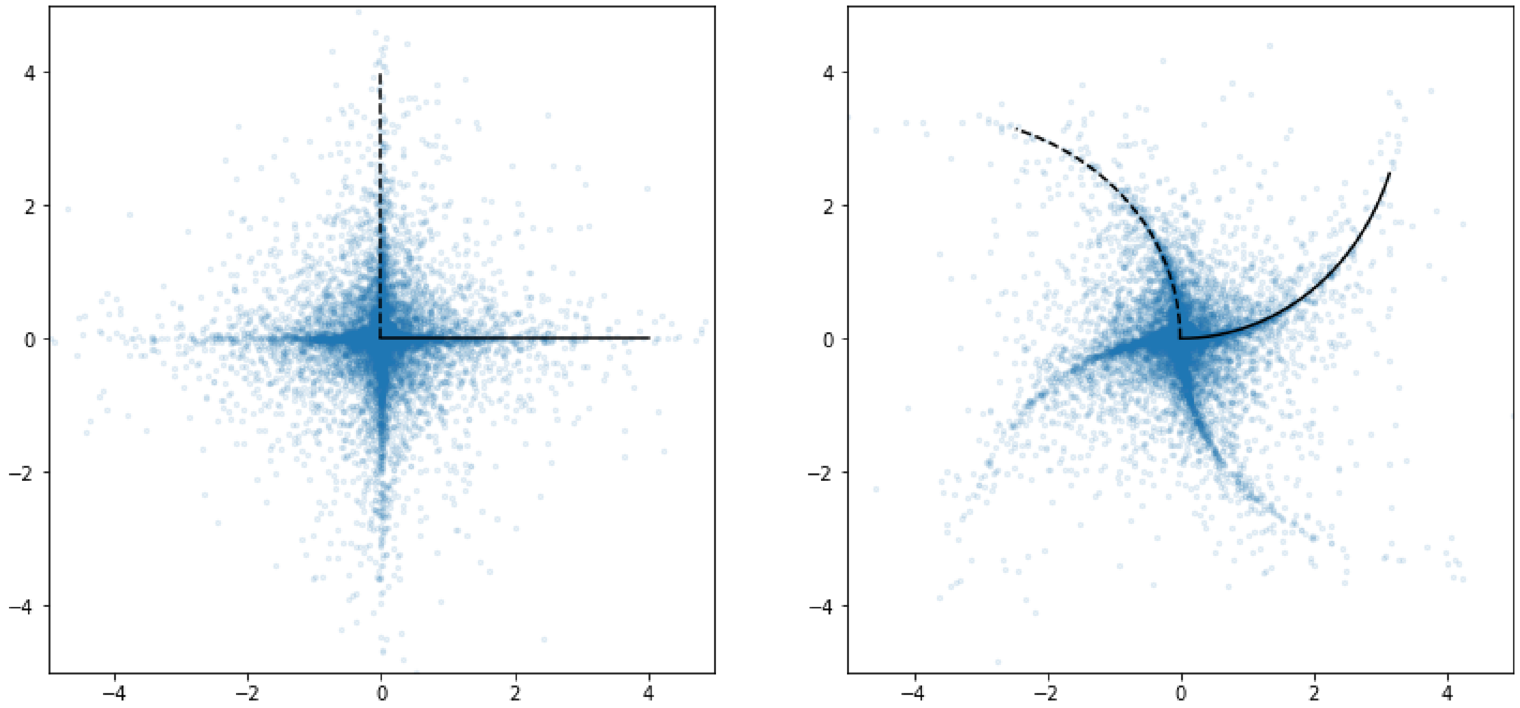
Latent space
(non-gaussian,
independent dimensions)

Invertible linear
transformation

Data space

Nonlinear ICA

- Replace the linear transformation with an invertible non-linear transformation



Latent space

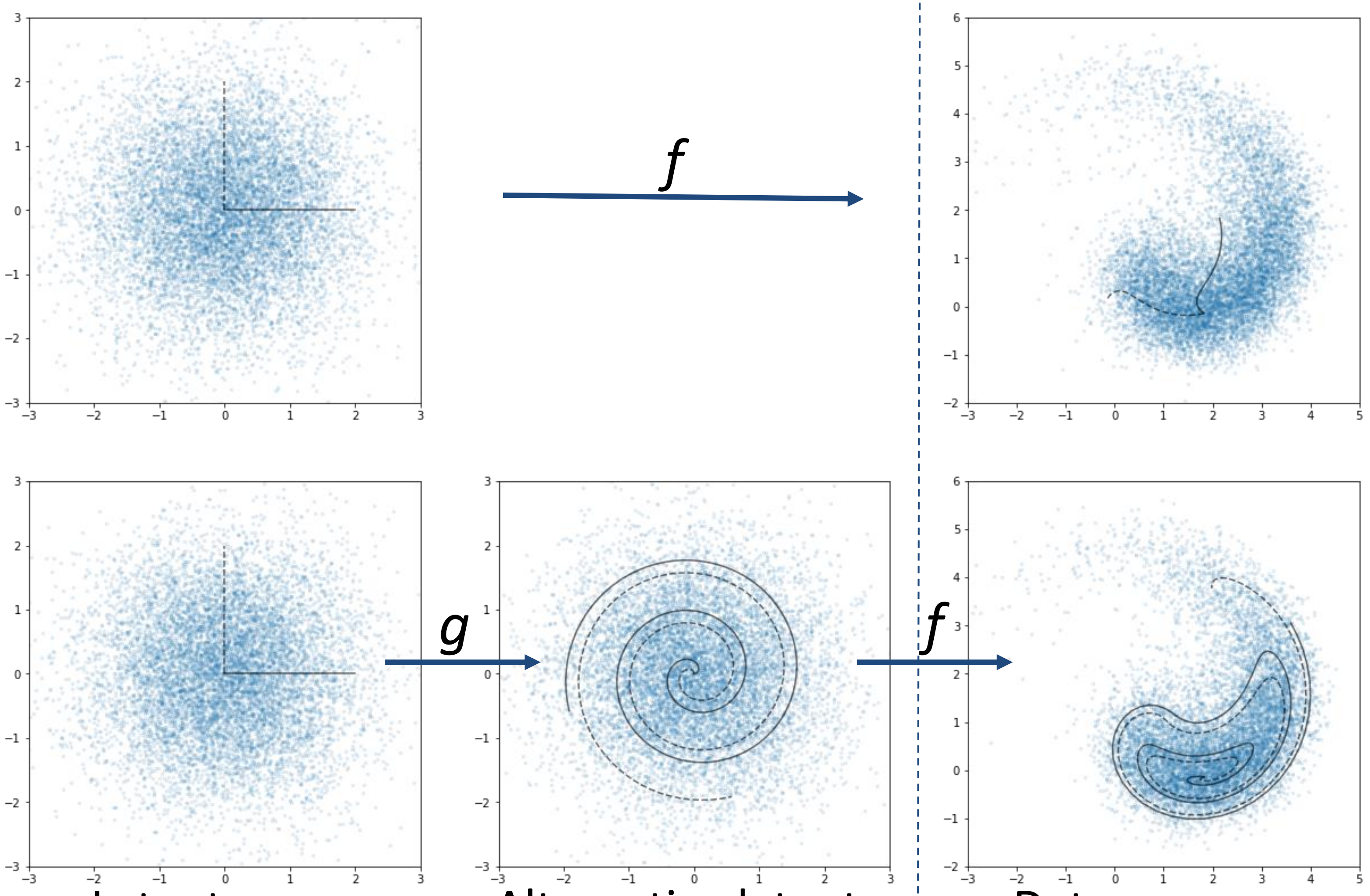
Invertible non-linear transformation

Data space

Non-linear ICA as a Disentanglement Method

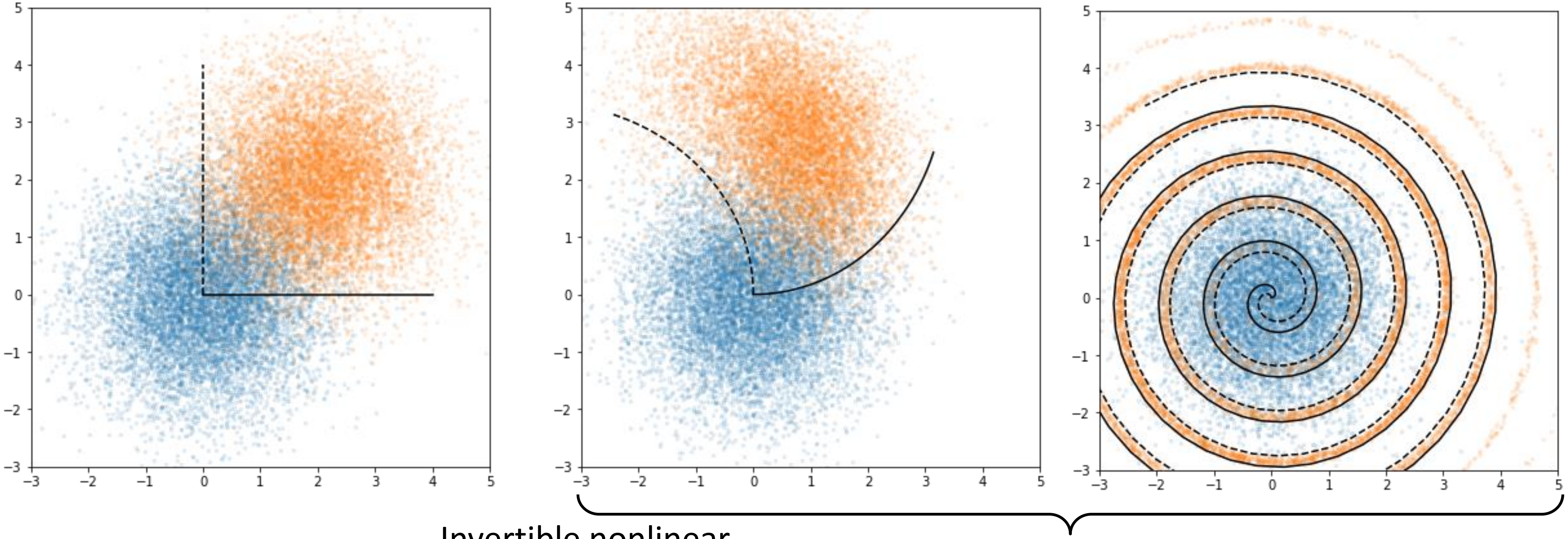
- Disentanglement: undo the non-linear mixing of given data, recover latent space
- This in general impossible: non-linear mappings are too flexible ⇨ ambiguity unresolvable
- Transformations $f(z)$ and $f(g(z))$ produce identical distributions

⇨ non-linear ICA is unidentifiable



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



Latent space
(mixture with conditionally independent dimensions)

Invertible nonlinear transformation

Alternative data spaces
(different solutions place the colors differently)

LM-INNs for Disentanglement

LM-INNs fulfill the theoretical assumptions of non-linear ICA

Important negative result [Hyvärinen & Pajunen 1999]:

Fully unsupervised *non-linear* disentanglement is impossible

General non-linear transformations are too powerful – can fit everything

Recent positive results: non-linear disentanglement becomes identifiable with additional conditioning information, e.g.

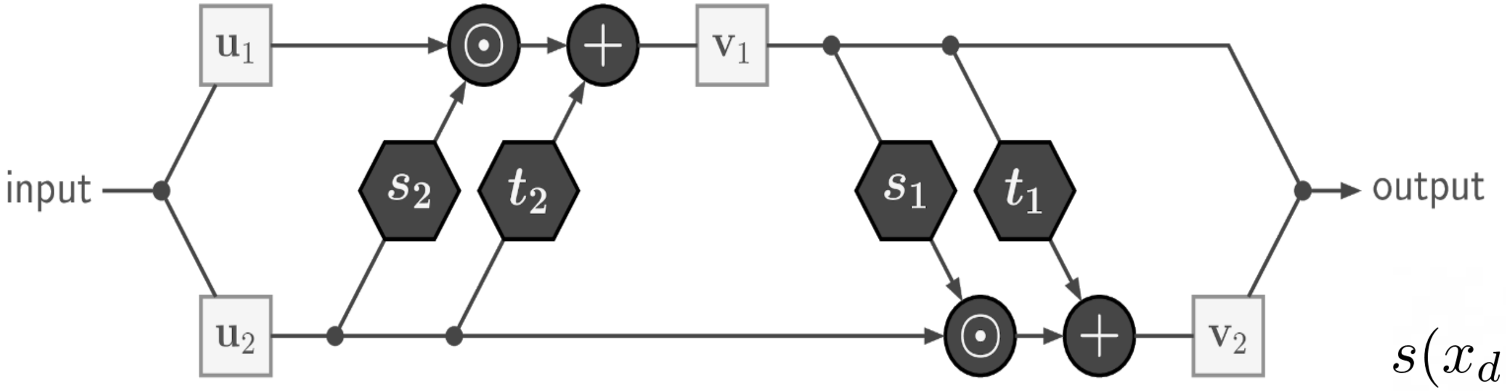
- Temporal relations [Hyvärinen & Morioka 2017, Hyvarinen, Sasaki, Turner 2018]
- Multi-modal observations [Gresele, Rubenstein, Mehrjou, Locatello, Schölkopf 2020]
- Class labels [our work]

⇨ Mathematical guarantees that non-linear ICA finds the *true generative factors* and the *true intrinsic dimension* in certain situations (generalizing this is a hot research topic).

Sorrenson et al. **"Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)"**, ICLR 2020.

General Incompressible-flow Networks (GIN)

- Modification of Real NVP coupling block architecture
  - Constrain the Jacobian to determinant 1
  - This differs from additive coupling (NICE): space can be scaled in some dimensions, when this is compensated for by a counter change in the remaining dimensions



$$\Rightarrow \sum_{i=1}^{d} s(x_i) = 0 \quad \Rightarrow \quad \log|J| = 0$$

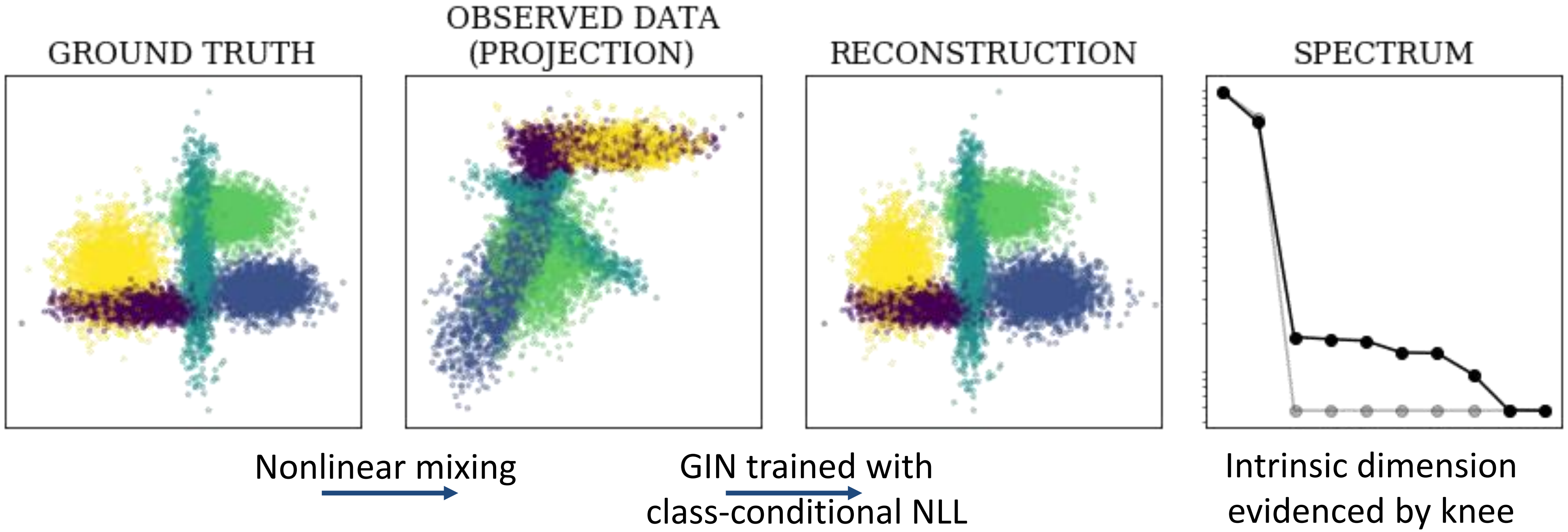
$$s(x_d) = -\sum_{i=1}^{d-1} s(x_i) \quad \text{or} \quad s(x_i) \leftarrow s(x_i) - \frac{1}{d}\sum_{i=1}^{d} s(x_i)$$

- Advantage:
  - Total "Variance" is preserved

  ⇨ Spectrum of latent variables can be sorted and interpreted as in PCA

Artificial Data Experiments

- True generative process:
  - 5 Gaussian mixture components (“class labels”), 2 meaningful dimensions, 8 noise dimensions
  - Mapped to 10-dimensional data space using a random non-linear transformation
- Task of the INN:
  - Determine that intrinsic dimension is 2
  - Recover the GMM within the meaningful dimensions, given the class labels



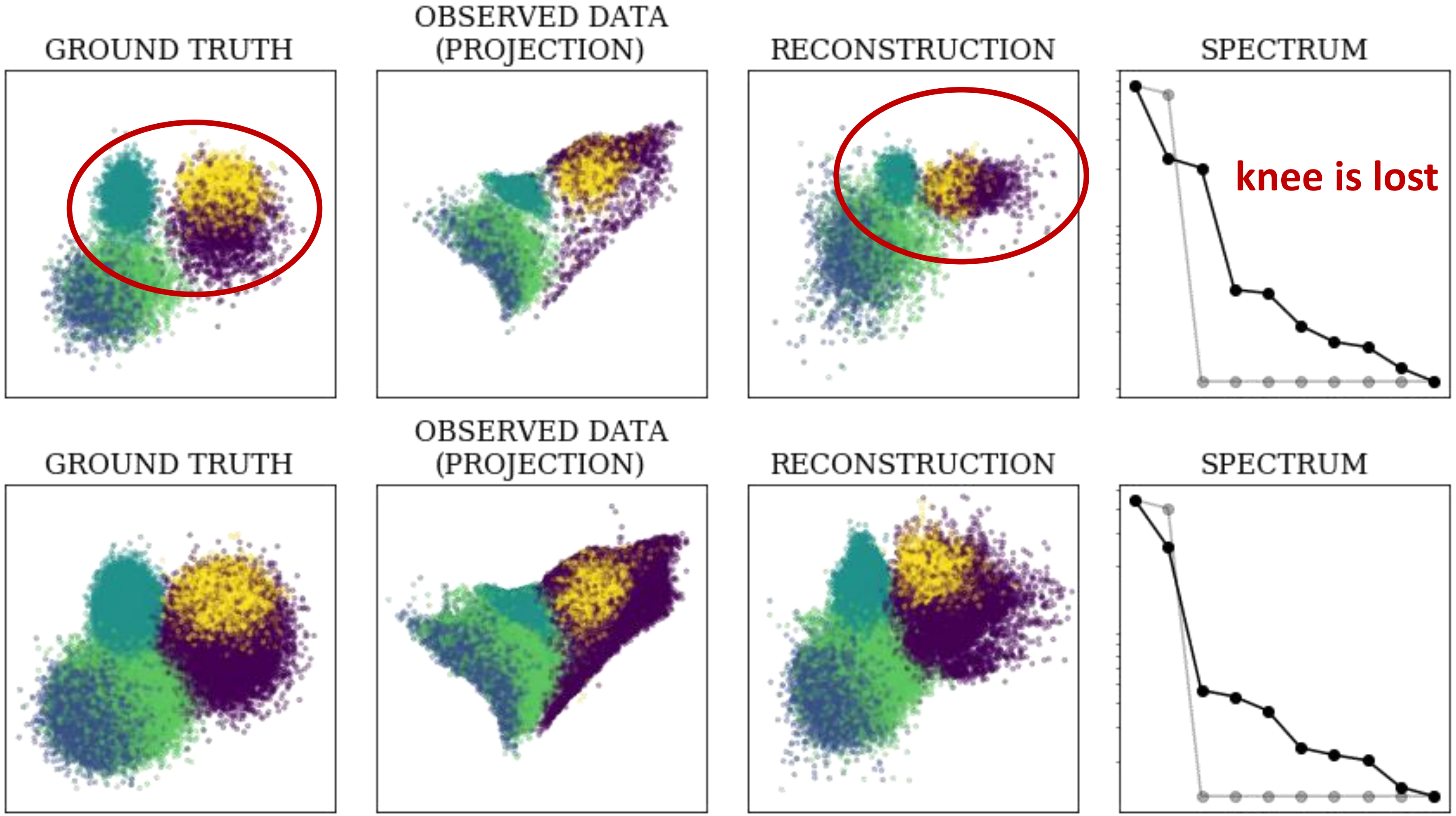
Sorrenson et al. **“Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)”**, ICLR 2020.

Artificial Data Experiments

- Intuition: works, because all clusters must be disentangled simultaneously
  - Breaks down, when clusters have no overlap: model transforms these clusters independently
  - Can be caused by lack of training data:

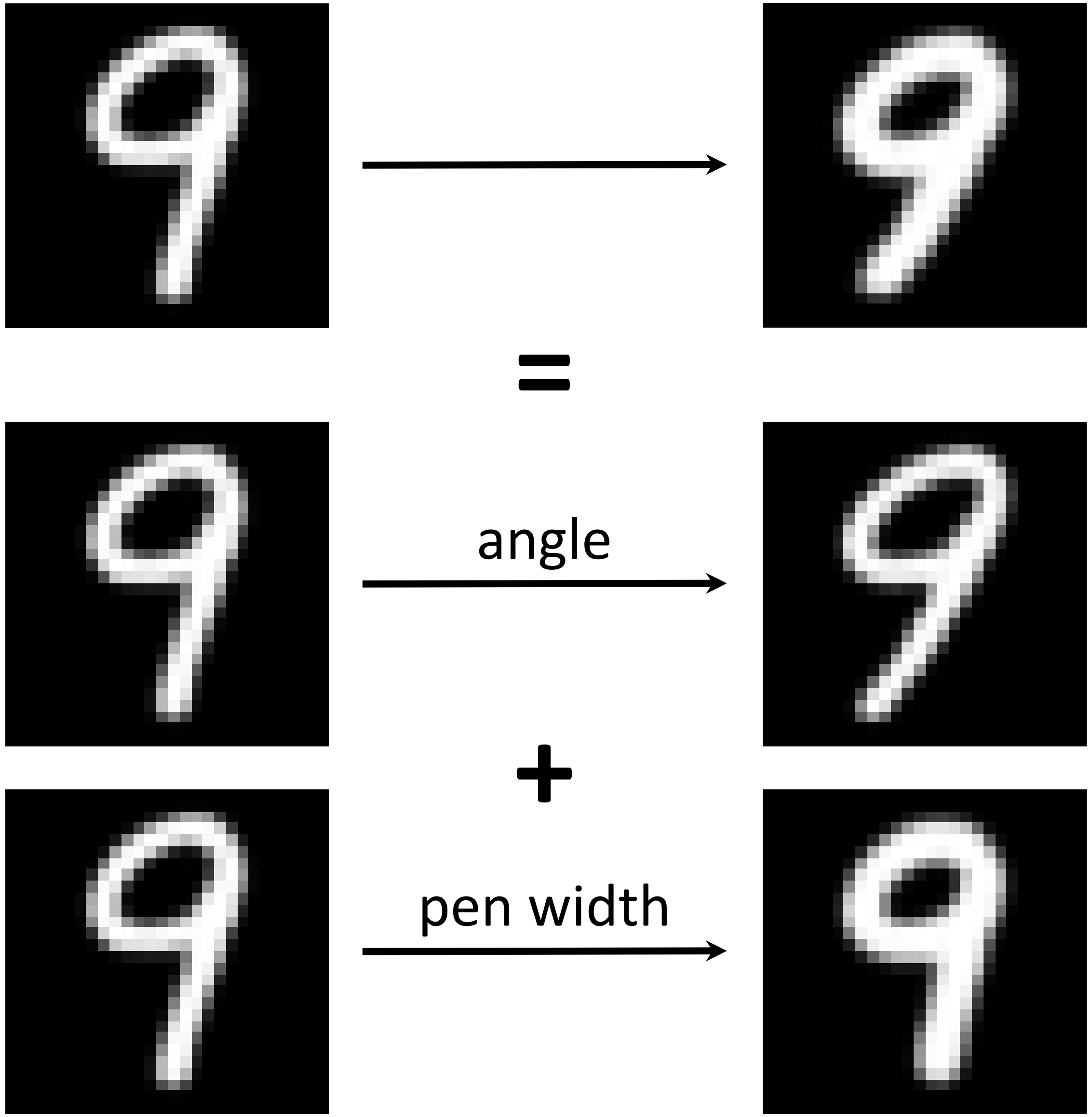
10^4 data points

10^5 data points



LM-INNs for Disentanglement

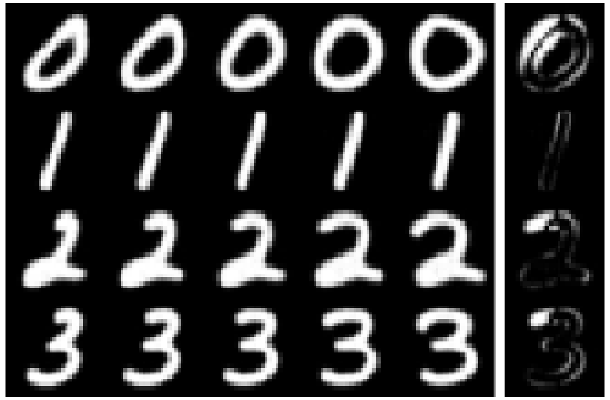
- Identify intrinsic factors of complicated data distributions, which intuitively explain variability
- Express complex/coordinated changes of the data as a combination of simple changes in the factors
- Example: EMNIST handwritten digits: latent factors are characteristics of handwriting styles



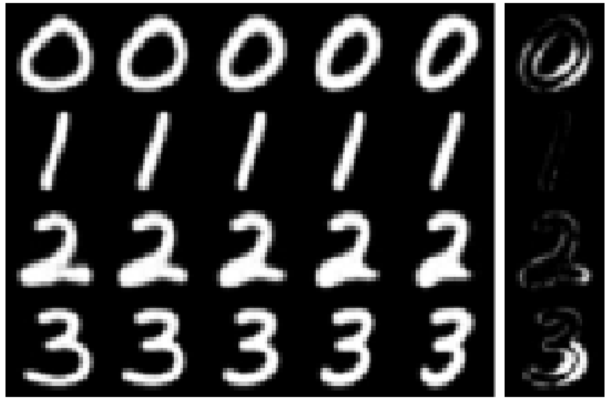
Sorrenson et al. **"Disentanglement by Nonlinear ICA with General Incompressible-flow Networks (GIN)"**, ICLR 2020.

Application to EMNIST

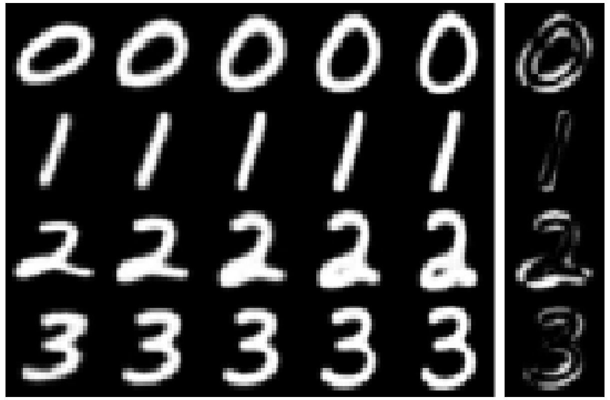
- First 8 latent variables control global properties
- Following 14 control local shape
- Remaining 762 have no visible effect



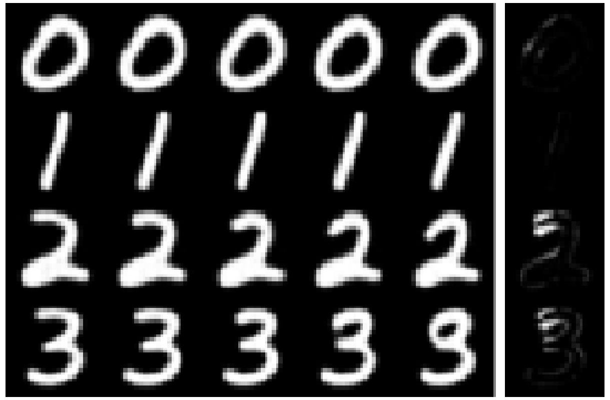
Var. 1: upper width



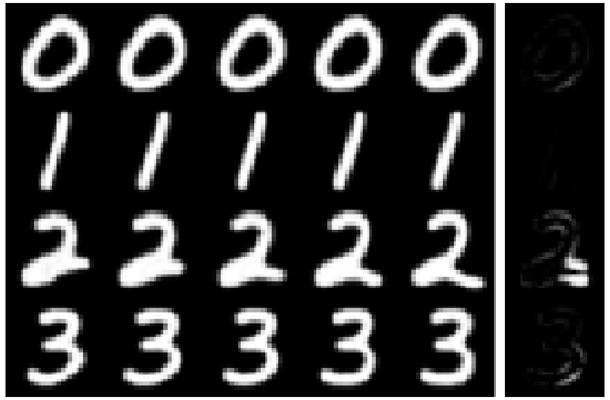
Var. 8: lower width



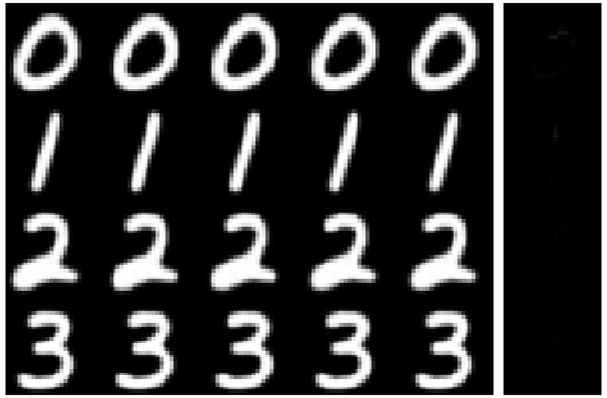
Var. 3: height



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



Var. 16: tail of 2

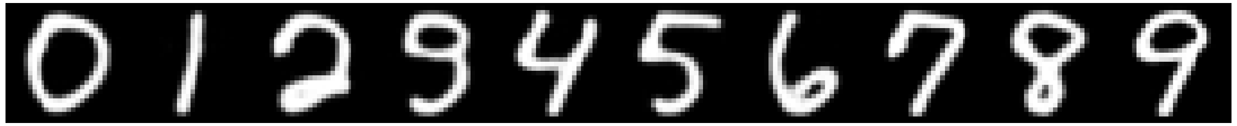


Var. 23: no effect

difference images

Application to EMNIST

Latent space interpolation



Independent effect of first 8 most significant latent dimensions

(animations not visible in PDF version)

IB-INNs – Building (more) interpretable models with INN-based generative classifiers

Ullrich Köthe
Visual Learning Lab, Heidelberg University
joint work with **Lynton Ardizzone, Radek Mackowiak, Jakob Kruse, Carsten Rother**

Tutorial „Normalizing Flows“ at CVPR 2021
June 2021

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:

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

- GCs promise to foster reliability and interpretability
  - uncertainty quantification, outlier detection, robustness against distribution shifts
  - discovery of meaningful features
  - *but:* predictive performance of GCs used to be unconvincing ⇨ discriminative classifiers (DCs) prevailed
- Old idea, but so far discriminative classifiers have much better performance

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

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

Generative aspect (minimize spurious information about x)

Trade-off parameter

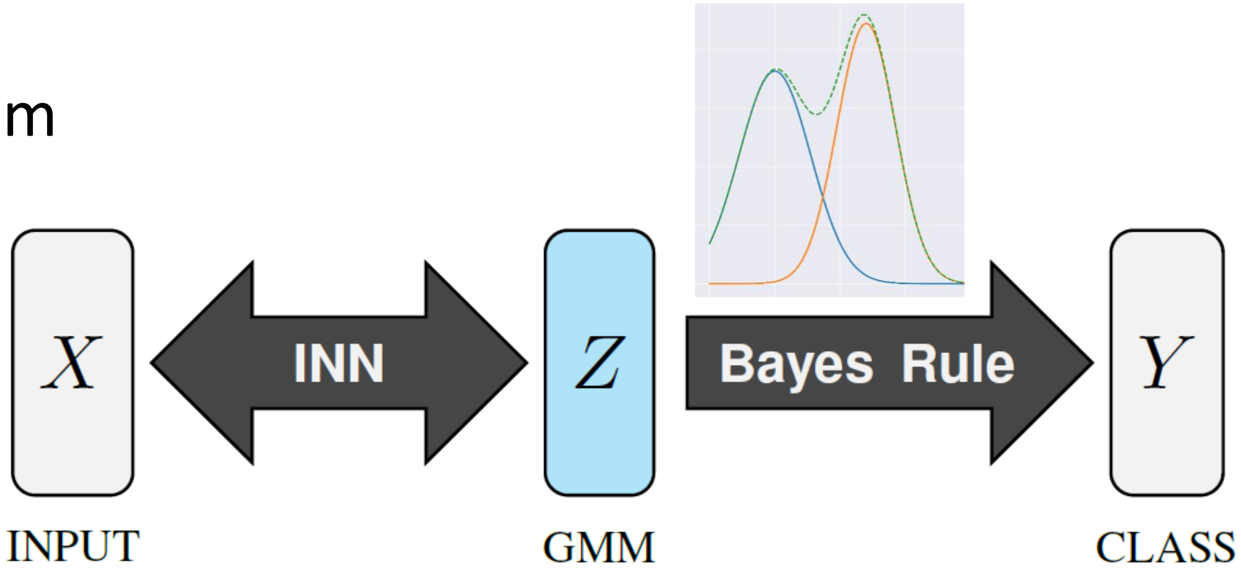
Discriminative aspect (maximize information about class labels)

with Mutual Information (MI) $I(y,z) = D_{\mathrm{KL}}\Big(p(y,z) \Big\| p(y)p(z)\Big)$

Tishby et al. **“The Information Bottleneck Method”**, ACCCC 1999.

IB-INNs: Generative Classifiers

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



$I(x;\ f(x'))$ $\quad$ $I(y;\ f(x'))$

- Problem: INNs are lossless encoders – where is the bottleneck?
  - Train the INN mapping $z = f(x)$ with noise augmented data $x' = x + \epsilon$:

$$\mathcal{L}_{IB} = \lim_{\epsilon \to 0} I\big(x; z_\epsilon = f(x')\big) - \beta \cdot I\big(y; z_\epsilon = f(x')\big)$$

  - Intuitively: noise ensures lossy encoding ⇨ prevents divergence of $I\big(x; z_\epsilon = f(x')\big)$
  - Surprisingly, mutual information $I(x; z_\epsilon)$ reduces to the usual maximum likelihood loss

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

Ardizzone et al. **“Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification”**, arXiv 2020.

IB-INN: Training an LM-INN as a Generative Classifier

LM-INN can approximate the IB loss arbitrarily well ⇨ IB-INN

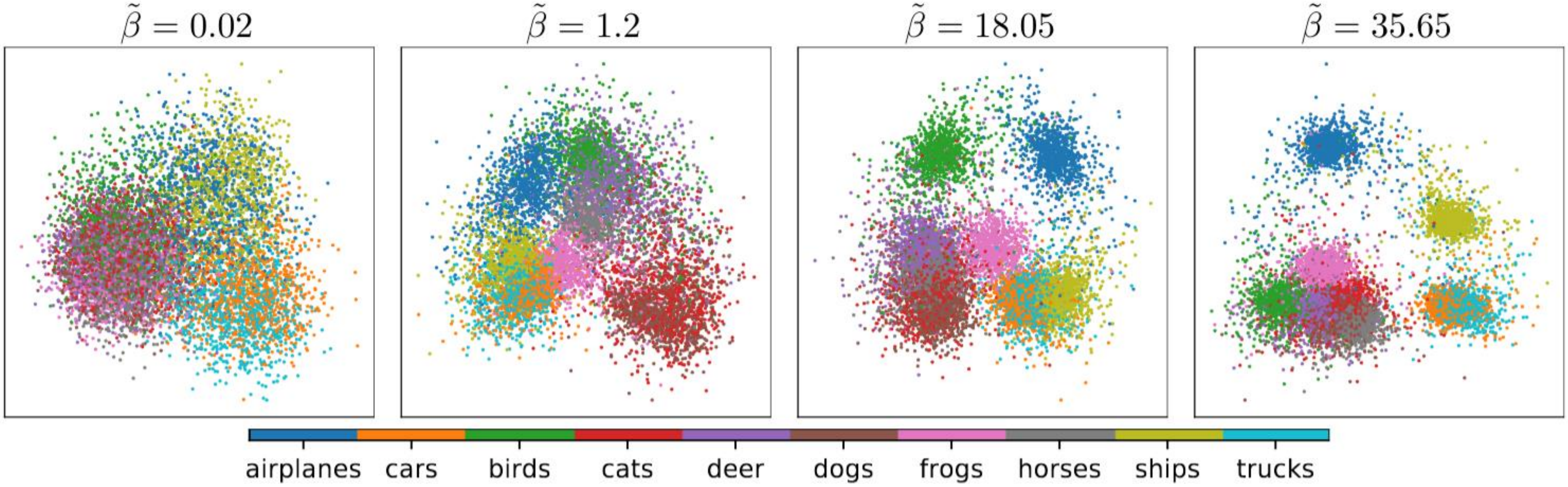
- Successfully trained on CIFAR-10 (10 classes, 32^2 images) and ImageNet (1000 classes, 224^2 images)
- Depending on β, the IB-INN emphasizes generative or discriminative performance
  - at $\beta = 1$, bits/dimension (=generative performance) comparable to a purely generative model
  - at β =32, test accuracy (=discriminative performance) comparable to a discriminative ResNet
  - ImageNet: good trade-off at $\beta = 8$

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”**, NeurIPS (oral) 2020.
Mackowiak, Ardizzone et al. **“Generative Classifiers as a Basis for Trustworthy Computer Vision”**, CVPR (oral) 2021.

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)



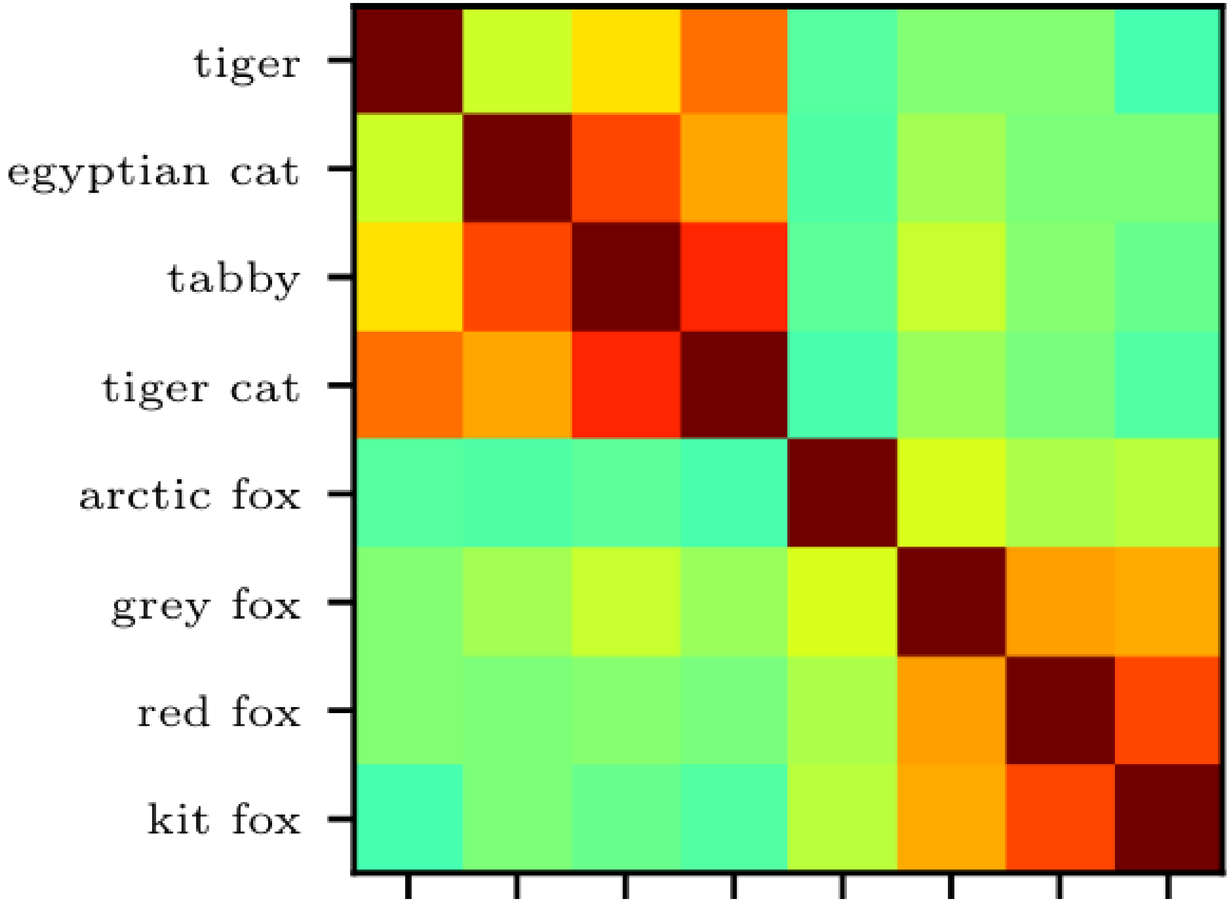
IB-INNs: Benefits of GC (1): Interpretability

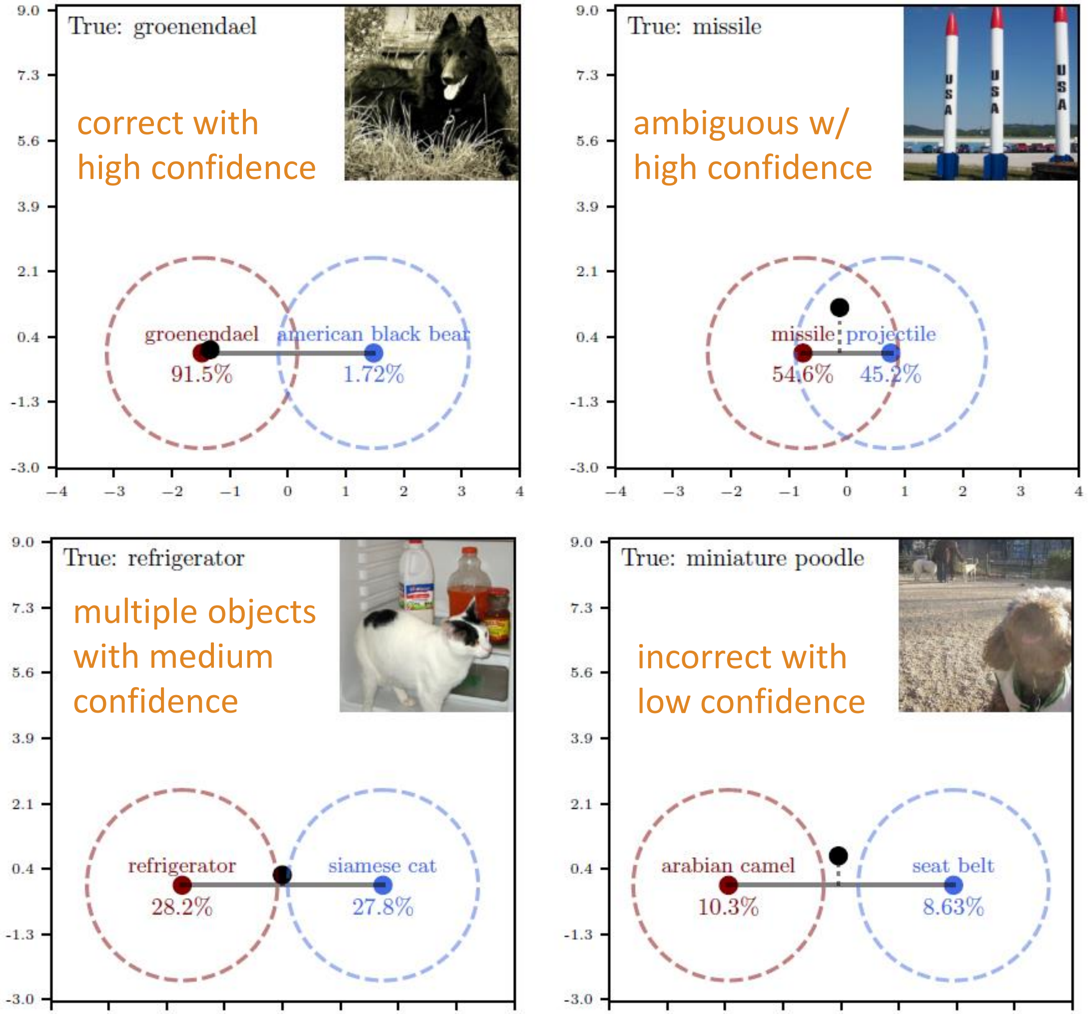
- Heatmaps for attention area of the most probable classes
  - Back-project relevant latent features to image space regions
  - Thanks to invertibility, the heat-maps represent the true decision process, not a post-hoc explanation

Mackowiak, Ardizzone et al. **"Generative Classifiers as a Basis for Trustworthy Computer Vision"**, CVPR (oral) 2021.

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

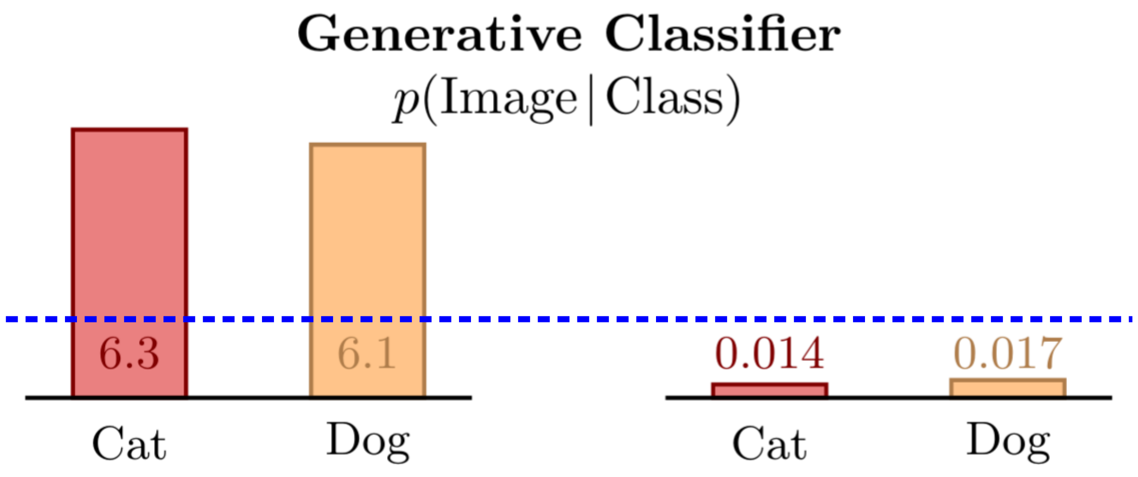
- Pairwise distances between class centers in *z*-space reflect class similarity and confidence
  - ImageNet examples





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
  - Many interesting open questions
    - Which outlier scores does IB-INN support? (e.g. typicality tests, WAIC, …)
    - What does it mean for an instance to be an outlier in *high dimensions*? (much of our intuition is based on low dimensional case and thus misleading)
    - Which latent re-parameterizations are sensitive for which type of outlier?

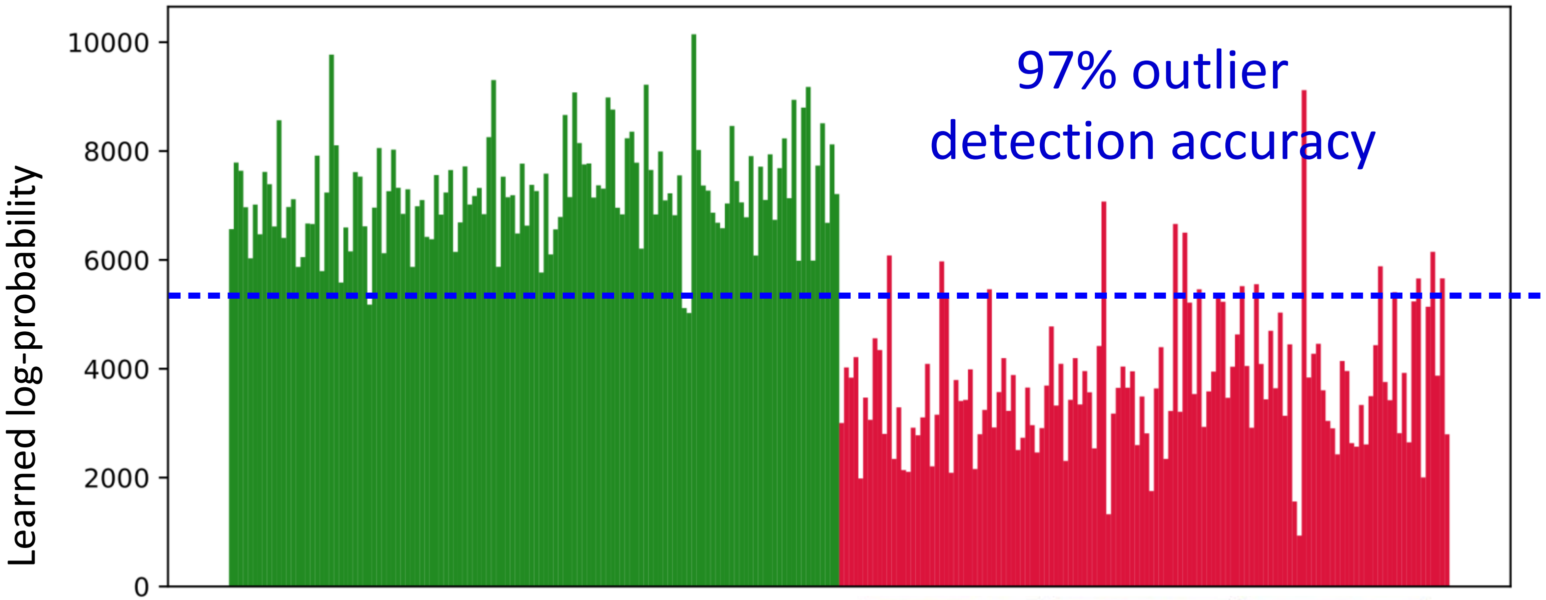


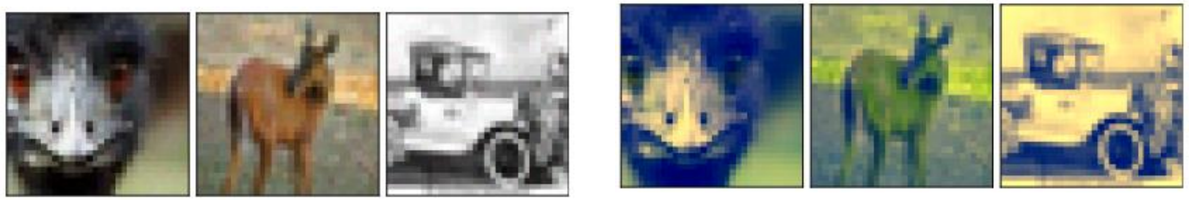
Normal input

OoD input

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

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

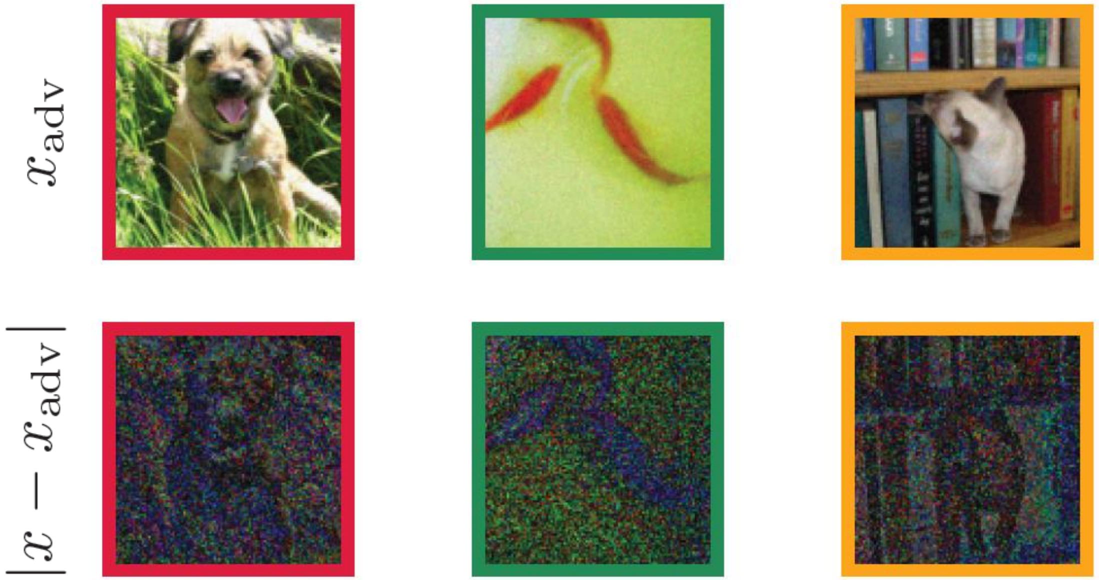




Inlier images (test set)

Same images, scrambled colors

Adversarial examples (ImageNet)



Minimal perturbations to get confident incorrect predictions

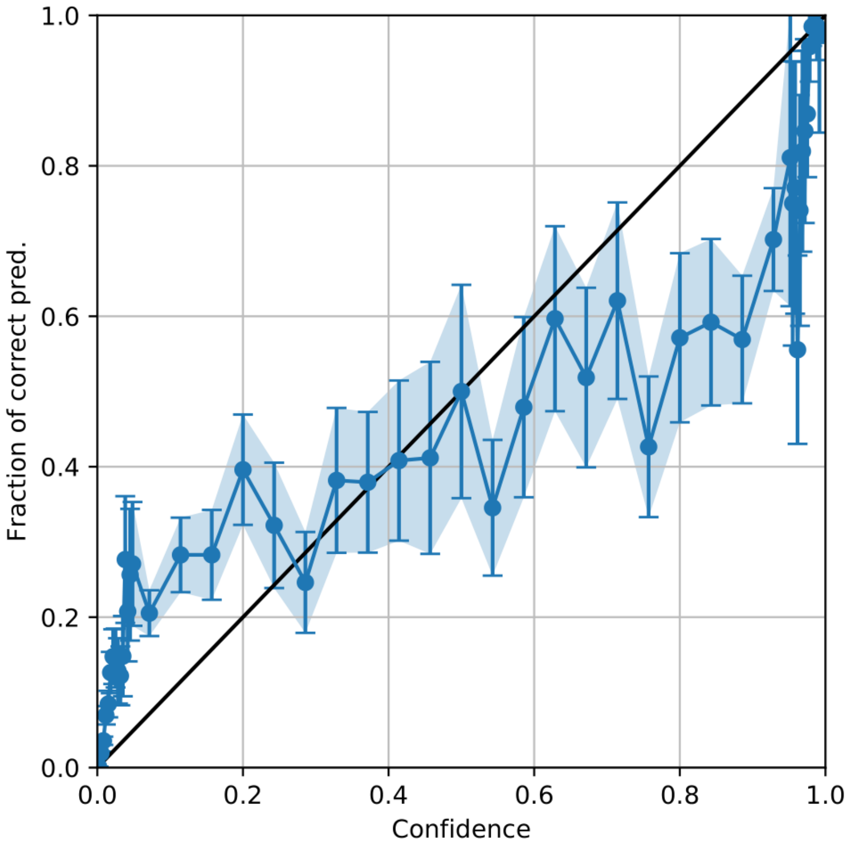
⇨ IB-INN improves adversarial robustness, but does not in itself solve the problem

Ardizzone et al. **"Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification"**, NeurIPS (oral) 2020.
Mackowiak, Ardizzone et al. **"Generative Classifiers as a Basis for Trustworthy Computer Vision"**, CVPR (oral) 2021.

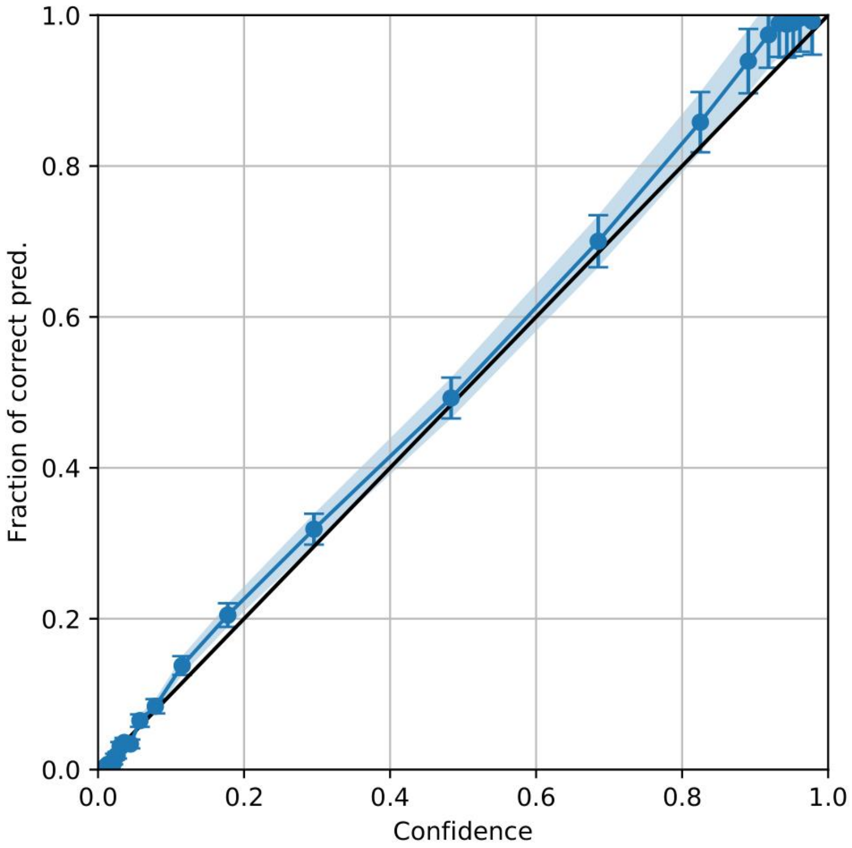
IB-INNs:
Benefits of GC (3): Uncertainty Calibration

- 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.

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
- 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
  - Apply to various problems in natural 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

Inst. for Theoretical Astrophysics, Uni Heidelberg:

Victor Ksoll, Ralf Klessen

Inst. for Environmental Physics, Uni Heidelberg:

André Butz, Florian Kleinicke

Psychologisches Institut, Uni Heidelberg:

Stefan Radev, Ulf Mertens, Andreas Voss

References

Public code of our INNs and papers in the FrEIA library: https://github.com/VLL-HD/FrEIA

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).

Adler, T. J., et al. „Out of distribution detection for intra-operative functional imaging", In: *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging and Clinical Image-Based Procedures* (pp. 75-82), arXiv:1911.01877 (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.: "Conditional Invertible Neural Networks for Diverse Image-to-Image Translation", GCPR 2020, 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", NeurIPS 2020 (oral), 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", SciPost Phys. 2020, arXiv:2006.06685 (2020).

Bieringer, S. et al.: "Measuring QCD Splittings with Invertible Networks", arXiv:2012.09873 (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).

Draxler, F., Schwarz, J., Schnörr, C., Köthe, U.: "Characterizing the Role of a Single Coupling Layer in Affine Normalizing Flows", GCPR (best paper finalist) (2020).

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

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

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

References

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

Kobyzev, S., Prince, JD., Brubaker, M.: “Normalizing Flows: An Introduction and Review of Current Methods”, arXiv:1908.09257 (2019).

Kruse, J., et al.: “Benchmarking Invertible Architectures on Inverse Problems”, ICML Workshop INNF, arXiv:2101.10763 (2019).

Kruse, J., Detommaso, G., Köthe, U., Scheichl, R. “HINT: Hierarchical Invertible Neural Transport for Density Estimation and Bayesian Inference”, AAAI 2021, arXiv:1905.10687 (2019)

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

Ksoll, V. et al.: “Measuring Young Stars in Space and Time”, part 1: arXiv:2012.00521 , part 2: arXiv:2012.00524 (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”, CVPR 2021 (oral), arXiv:2007.15036 (2020).

Müller, J., Schmier, R., Ardizzone, L., Rother, C., Köthe, U. (2020). “Learning Robust Models Using The Principle of Independent Causal Mechanisms” arXiv:2010.07167 (2020).

Nölke, JH. et al.: “Invertible Neural Networks for Uncertainty Quantification in Photoacoustic Imaging”, arXiv:2011.05110 (2020).

Papamakarios, G. et al.: “Normalizing Flows for Probabilistic Modeling and Inference”, JMLR, 22(57):1-64, arXiv:1912.02762 (2021).

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

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

Radev, S., et al.: “Model-based Bayesian inference – an application to the COVID-19 pandemics in Germany”, arXiv:2010.00300 (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. Communic., Control, Computing, arXiv:0004057 (1999).