Interpretable Machine Learning and Hyperbolic Geometry

Ullrich Köthe

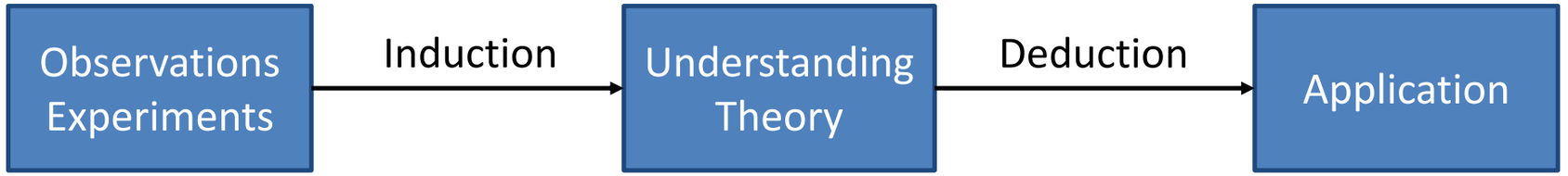
Visual Learning Lab, Heidelberg University

joint work with **Lynton Ardizzone, Jakob Kruse, Stefan Radev, Jens Müller, Felix Draxler, Peter Sorrenson, Carsten Rother**

Seminar Hyperbolic Geometry & Data Science, December 2020

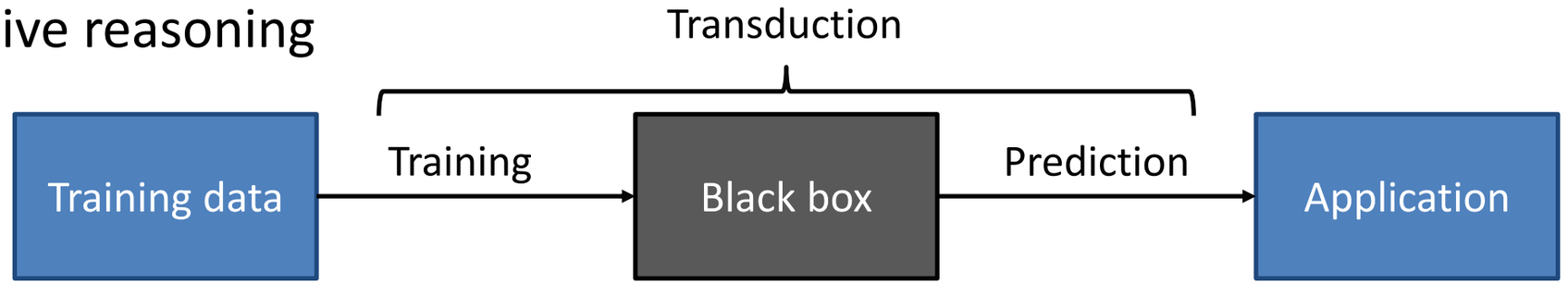
Inductive vs. Transductive Reasoning

- Traditional science: inductive reasoning



  - Problem: did not work well for machine learning until recently
    - available models were too weak to theoretically explain complex phenomena
    - ML community gave up on it around 1990

⇨ Transductive reasoning

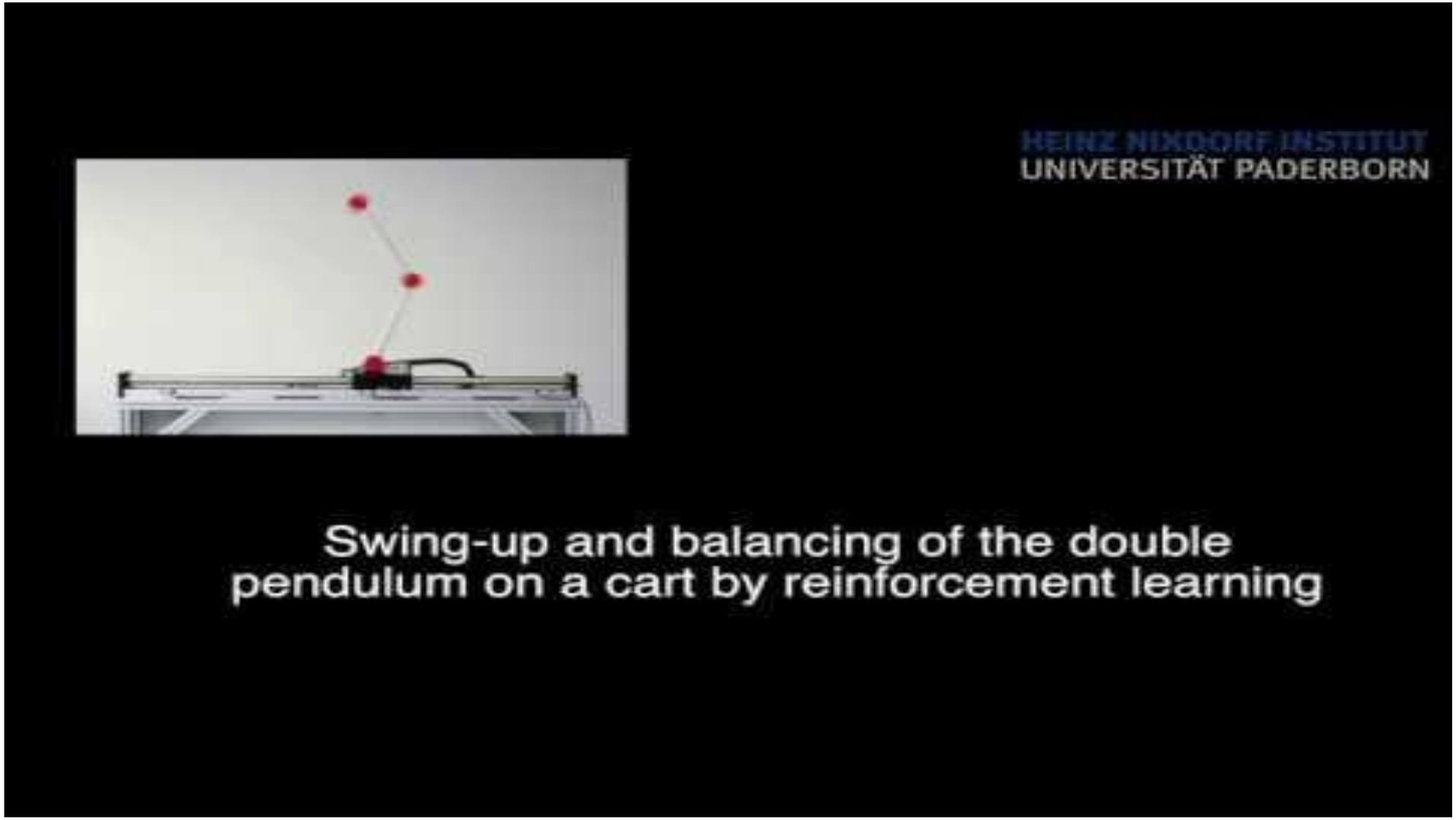


  - Why trying to understand the world, when one is only required to make good predictions?
  - Such black-box models are very general and work surprisingly well in practice
  - Most ML successes rest on transductive approach

Inductive Solution of Inverted Triple Pendulum: Optimal Control based on Equations of Motion

Control Engineering Group - TU Ilmenau (2018)

Transductive Solution of Inverted Double Pendulum: Reinforcement Learning



Hesse et al. (2018)

The Brain's Solution of Complex Balancing: Transductive or Inductive?

Generative Modelling

- Problem: transductive models are hard to interpret, explain, validate
  - not acceptable for applications with critical consequences
  - nor acceptable if ML shall be used as a scientific research tool
  - the return to inductive modelling is now within reach!

⇨ Generative modelling

- Transductive models: to predict y from x, learn the posterior distribution $p(y \mid x)$
- Generative models turn the problem around: learn the data generating process, i.e. likelihood $p(x \mid y)$
- Solve the original prediction task via Bayes theorem

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

- This is much more difficult and requires **insight**, i.e. theoretical understanding

Feynman: “What I cannot create, I do not understand.”

Neural Networks

- Standard definition of a neuron: $\mathbb{R}^D \to \mathbb{R}$: $z' = \phi(\boldsymbol{w}^T \boldsymbol{z} + b)$
  - weights $\boldsymbol{w}$, bias b, non-linear activation function ϕ, e.g. $\phi(t) = \tanh(t)$ or $\phi(t) = \max(t, 0)$ (ReLU)
- Fully connected networks: combine neurons in parallel and in series (= layers)
  - $f: \boldsymbol{z}_0 \in \mathbb{R}^D \to \boldsymbol{z}_L \in \mathbb{R}^M$: recursive definition $\boldsymbol{z}_l = \phi_l(\boldsymbol{W}_l^T \boldsymbol{z}_{l-1} + \boldsymbol{b}_l)$ with weight matrices $\boldsymbol{W}_l$ and element-wise application of ϕ_l
  - deep learning: large number L of layers (10 … 1000)
- Convolutional networks: weight matrices $\boldsymbol{W}_l$ are Toeplitz matrices
  - Additionally: some layers execute “max pooling”: reduce spatial resolution by ½ in every direction by taking the maximum of certain subsets of the input $\boldsymbol{z}_{l-1}$ (e.g. of 2x2 blocks of pixels)
  - Image classification networks: convolutional and pooling layers followed by fully connected layers
  - Image segmentation networks: only convolutional and pooling layers
- Most successful networks are transductive (“discriminative models”)

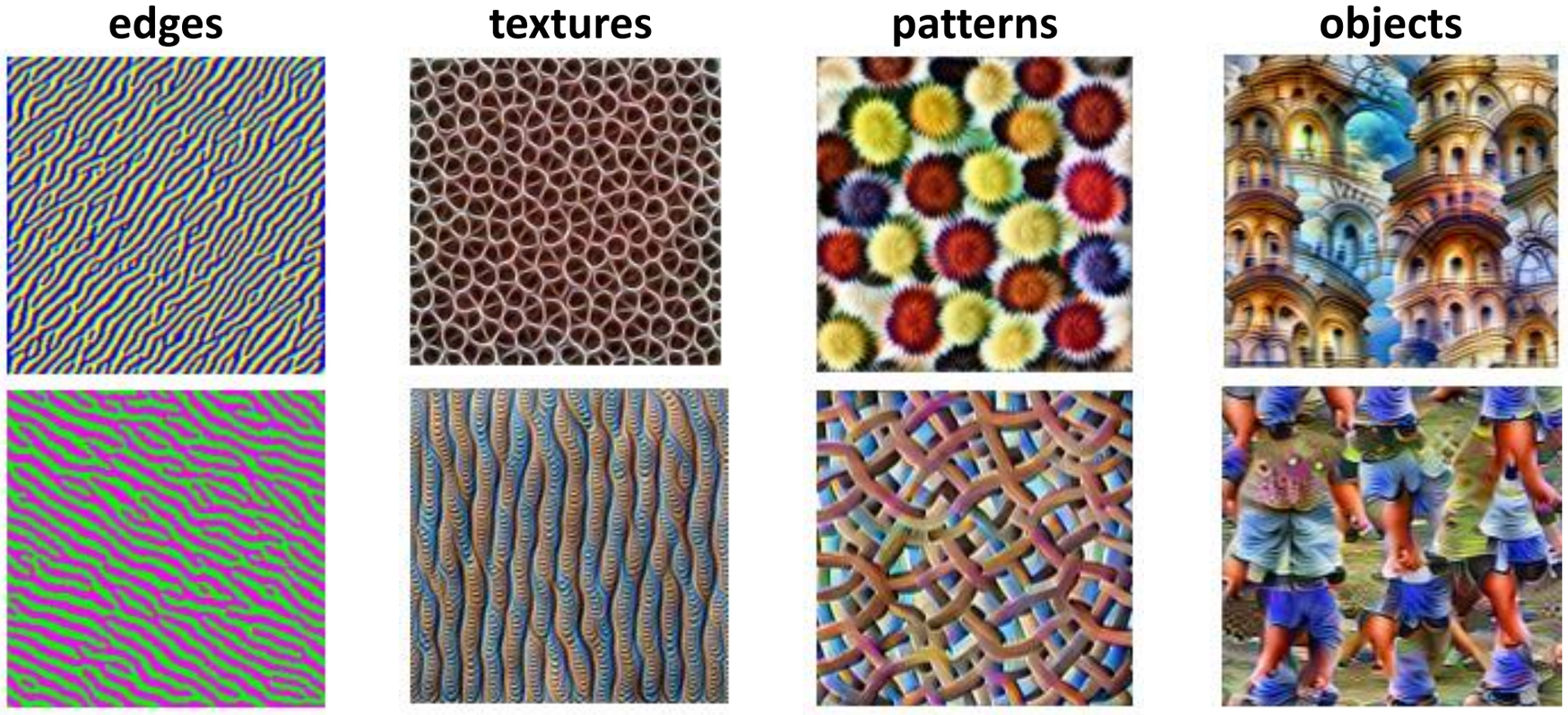
Deep Learning Example: Semantic Segmentation

- Neural network classifies each pixel as „Street“, „Sidewalk“, „Person“, „Car“ etc.
  - Prerequisite for autonomous driving, but not as long as it is an unverifiable black-box

Pohlen et al. (2016)

Opening-up the Black-Box in Retrospect

- Neural networks learn suitable features for the task
  - better than traditional hand-crafted features
  - early layers detect local edges, later layers entire objects
  - feature visualization shows what each layer is looking for:



Ohla et al.(2017)

Opening-up the Black-Box in Retrospect

- Emergent structure evolves in the feature spaces
- Can identify linear “concept axes”
- Extrapolation along a concept axis changes image in well-defined way
  - transform data $x \rightarrow z$
  - $z' = z + \tau \cdot \text{concept}$
  - reconstruct $z' \rightarrow x'$

- Active concepts in z serve as explanations

Upchurch et al.(2016)

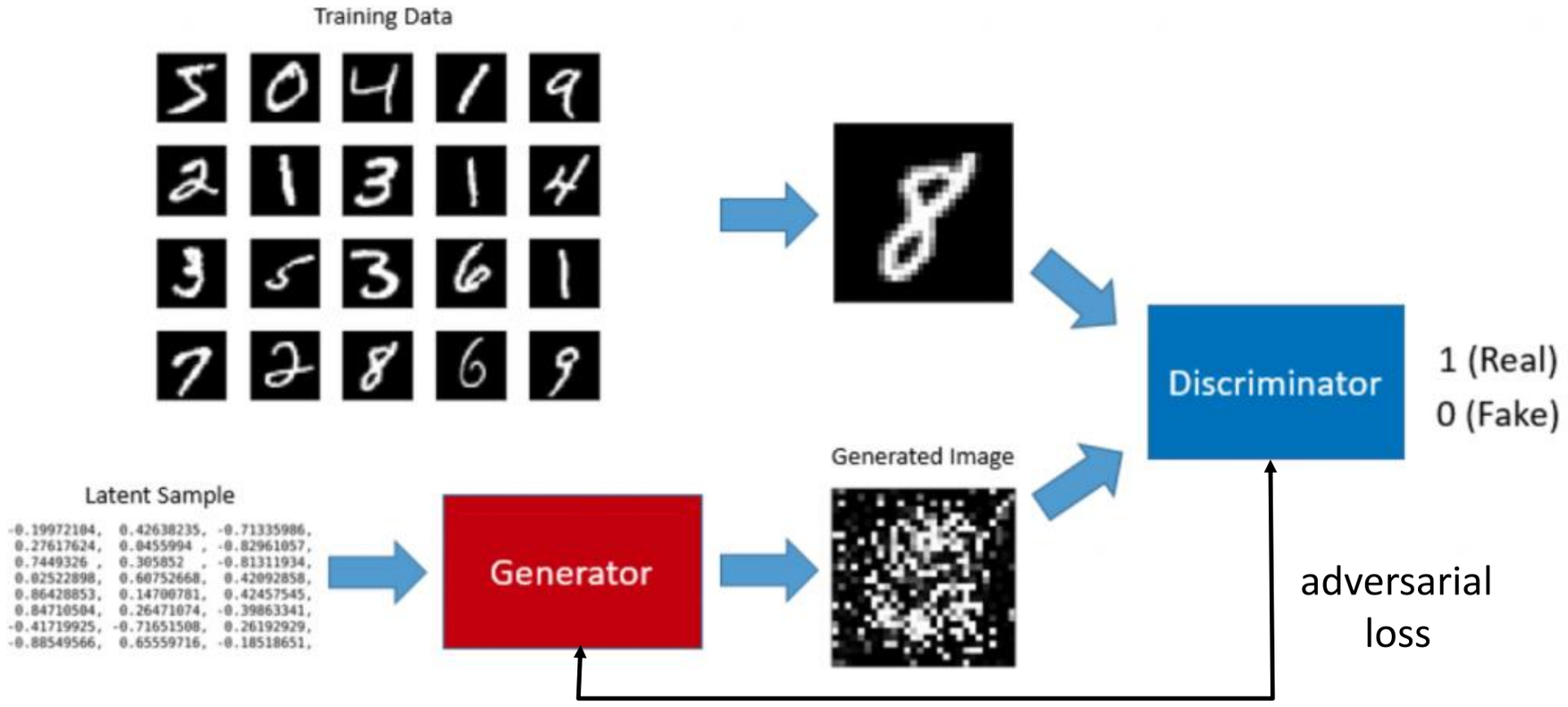
Generative Modeling

- Given features, learn to generate corresponding data

NVIDIA (2019)

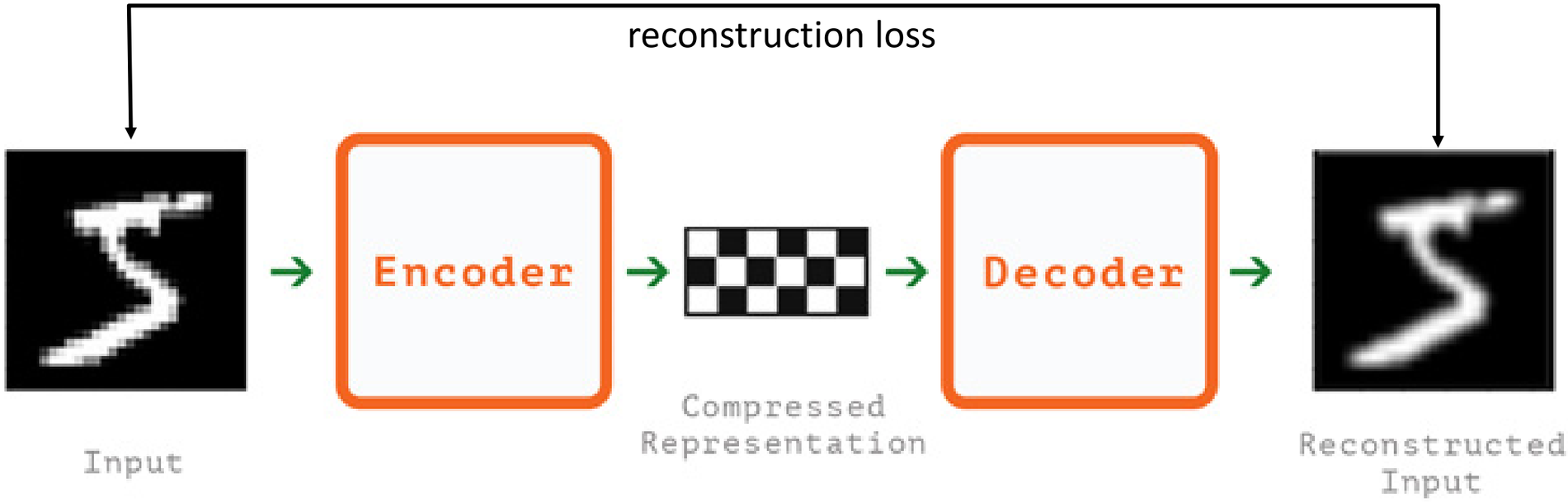
Deep Generative Modelling

- Generative Adversarial Networks (GANs):
  - two player game: generator learns to create realistic images
    discriminator learns to recognize synthetic images
  - converge to a state where synthetic and real images are indistinguishable



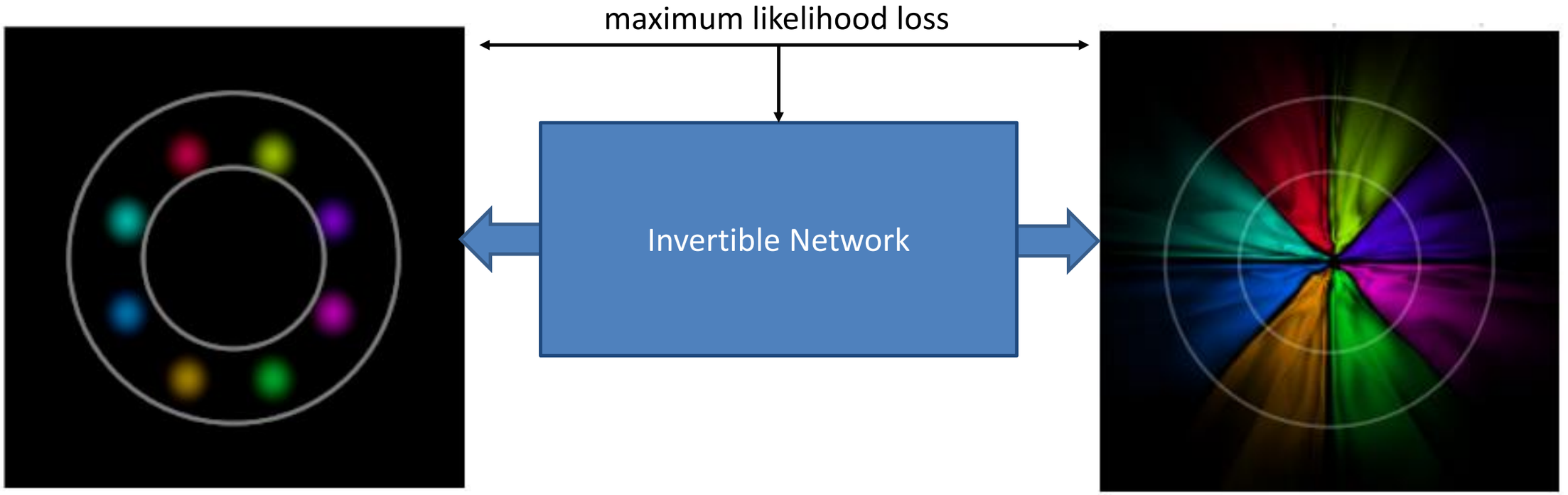
Deep Generative Modelling

- Autoencoder
  - learn a lossy compression through a latent bottleneck
  - converge to maximum reconstruction quality by keeping only the relevant information in the bottleneck



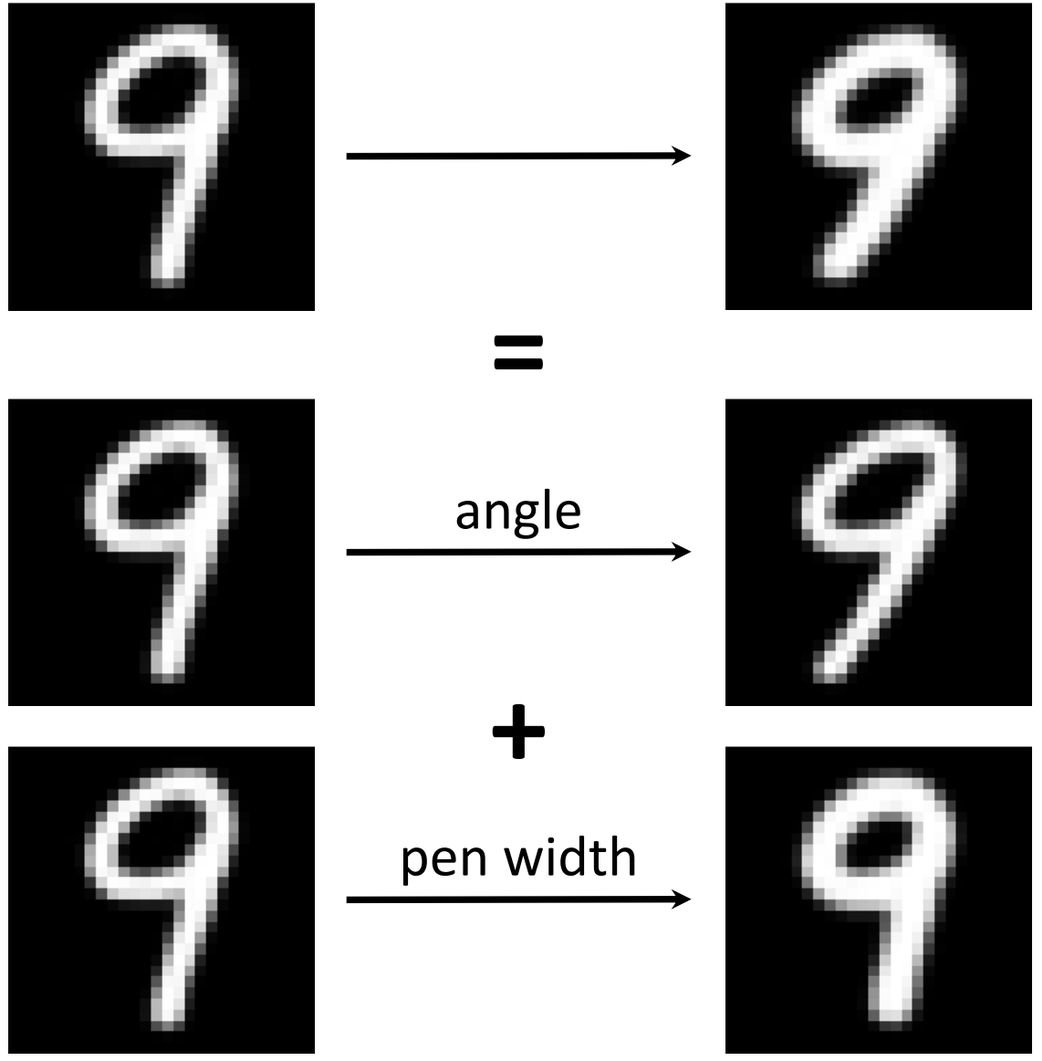
Deep Generative Modelling

- Normalizing flows
  - learn an invertible transport map between target distribution in data space and a simple latent base distribution
  - converge to maximum reconstruction quality by keeping only the relevant information in the bottleneck



Generative Modelling for Interpretable Machine Learning

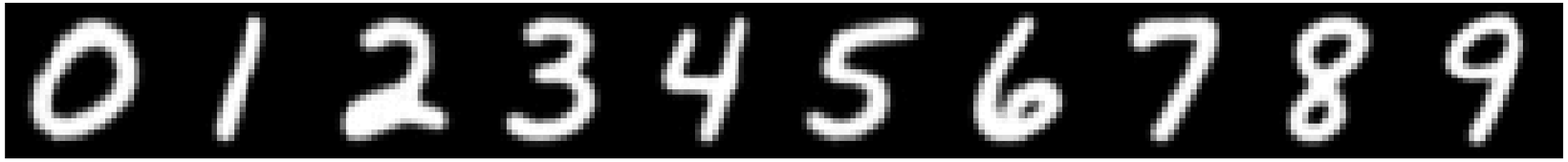
- Idea: control training such that the latent space evolves **interpretable structure**
  - Happens emergently in classical training, but can we do better?
- Example 1: disentanglement
  - each latent variable has well-defined meaning: encodes a single isolated data property
  - complex transformations of the real data can be expressed by simple latent superpositions
  - analogy in physics: finding a good coordinate transformation is key to the analytic solution of many problems



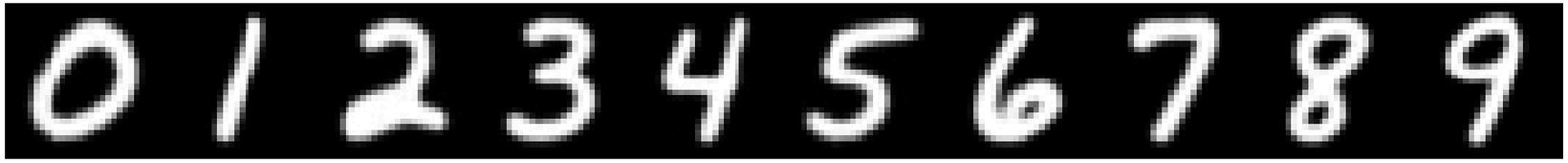
Disentanglement of MNIST using Invertible Neural Networks

- First 8 latent variables control global properties

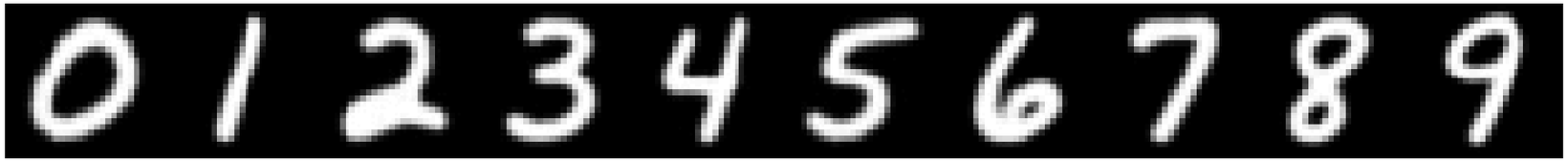
variability of handwriting in MNIST digits dataset



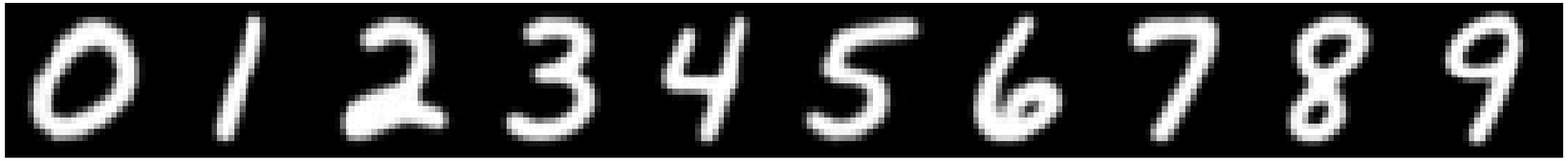
Slant



Height



Width of top

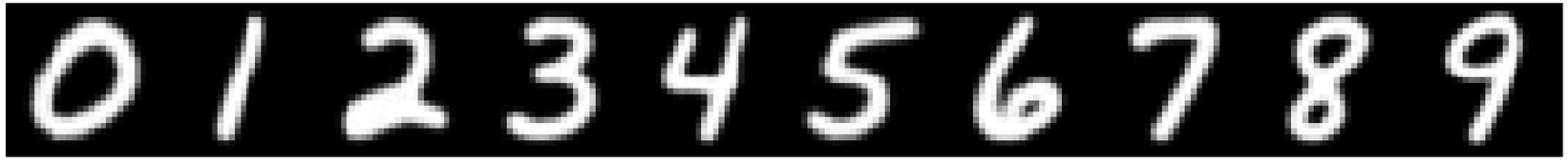


Width of bottom

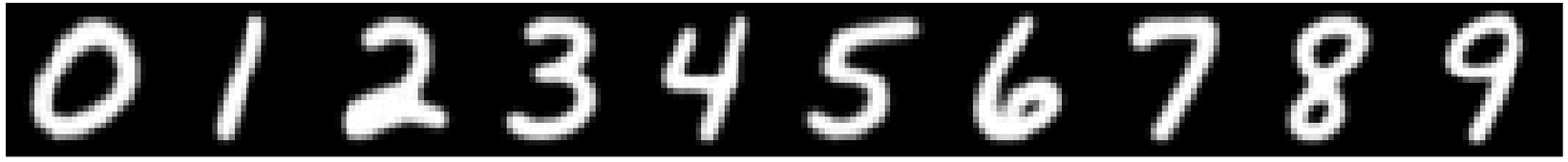
Disentanglement of MNIST using Invertible Neural Networks

- First 8 latent variables control global properties
- Following 14 control local shape

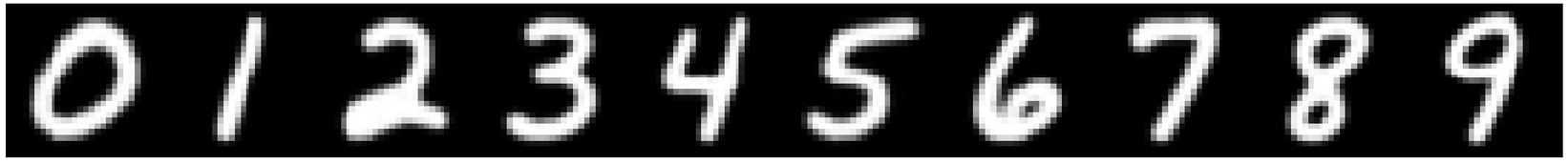
variability of handwriting in MNIST digits dataset



Openness of lower loop



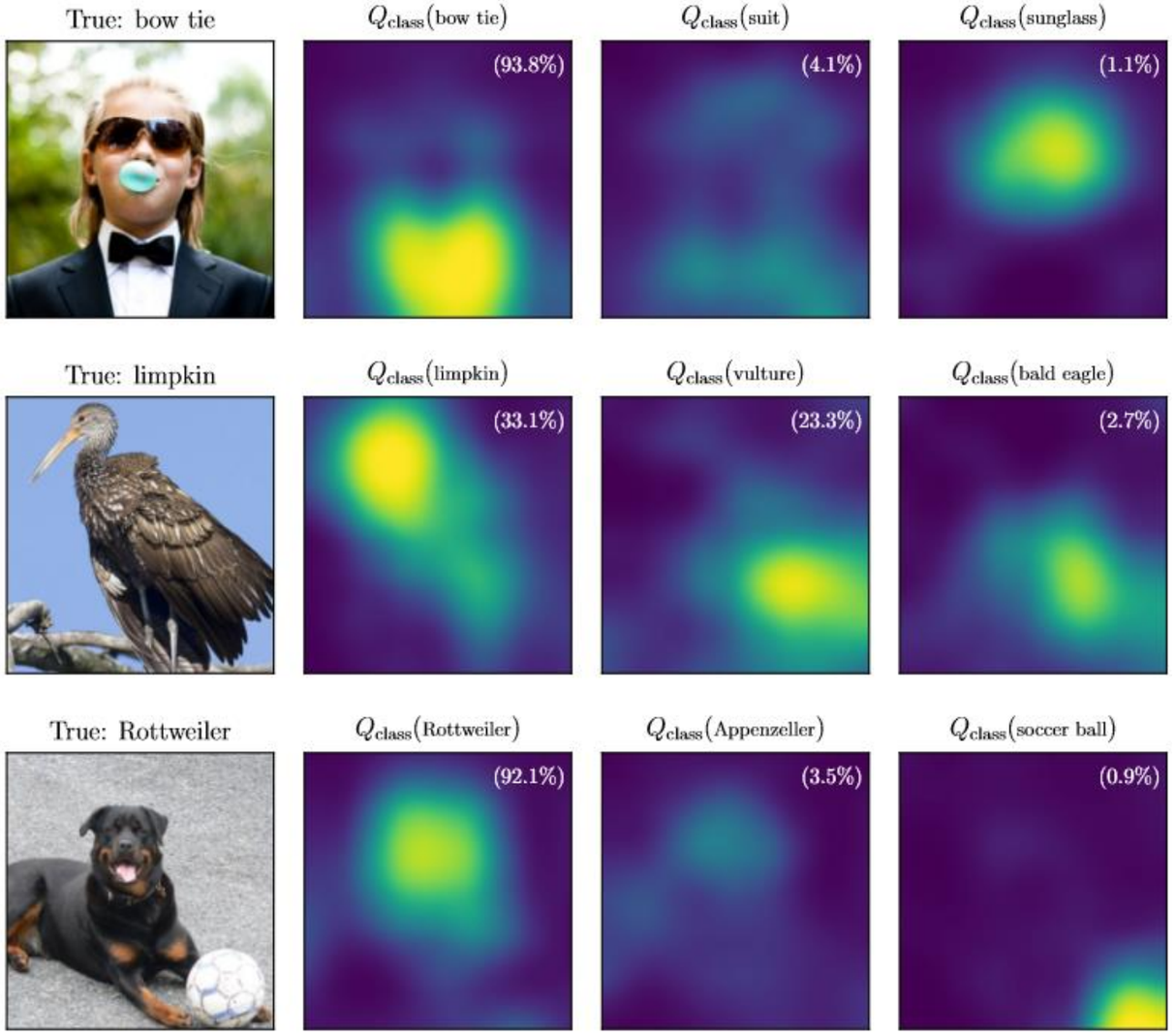
Extension to top right



Tail of 2

Generative Modelling for Interpretable Machine Learning

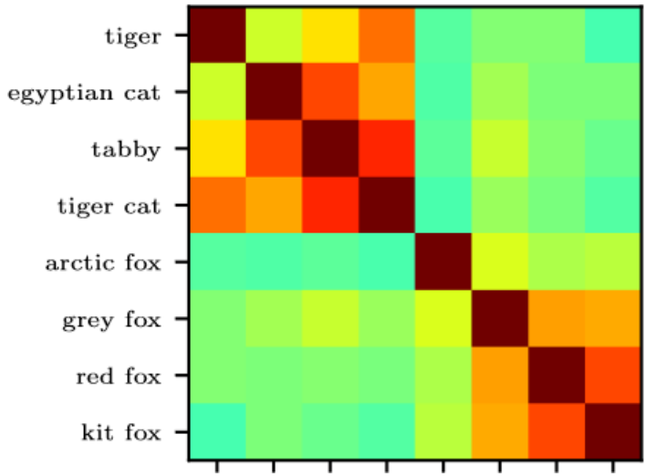
- Idea: control training such that the latent space evolves **interpretable structure**
  - Happens emergently in classical training, but can we do better?
- Example 2: Class Activation Mapping
  - Map latent interpretations back to data space
  - Heat maps: which input pixels are relevant to particular latent interpretations?
  - Visualize plausibility of interpretations and detect overfitting

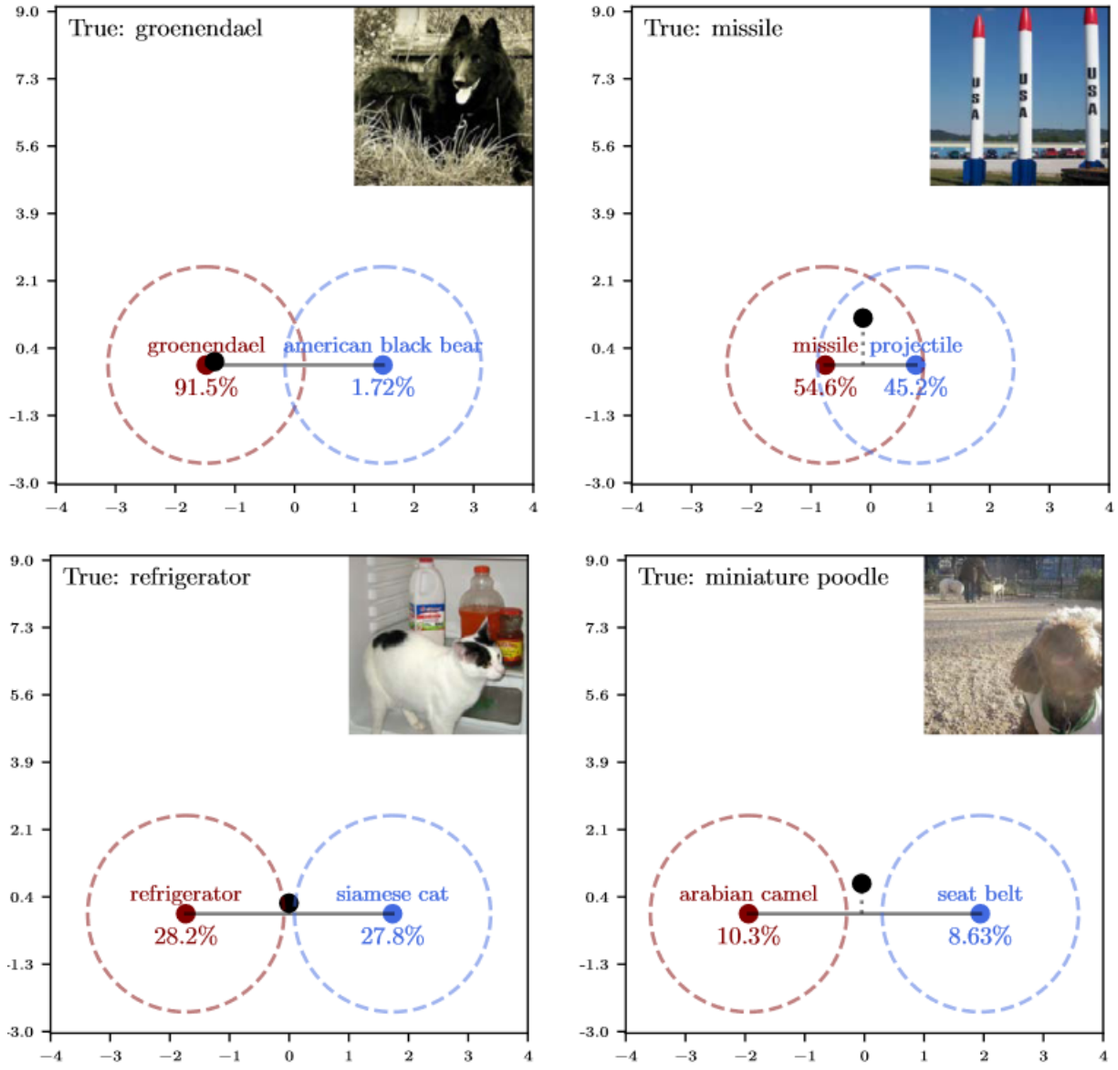


Generative Modelling for Interpretable Machine Learning

- Idea: control training such that the latent space evolves **interpretable structure**
  - Happens emergently in classical training, but can we do better?

- Example 3: Metric learning
  - Simple metric in latent space encodes complicated similarity relations in data space





Two Perspectives on Network Interpretability

1. External perspective: What black-boxes do we want for interpretable ML?
   Assume that networks can represent any desired mapping $\boldsymbol{z} = f_\theta(\boldsymbol{x})$
   - What types of mappings are especially suitable for interpretability?
   - Which properties should these mappings have (concept disentanglement, group equivariance, …)
   - How can we quantify the uncertainty of such mappings (Bayesian posteriors, outlier detection, …)
   - How can network mappings serve as a tool for scientific discovery?

2. Internal perspective: How to achieve the desired mappings?
   Open-up the black-box
   - What network architectures realize interesting interpretable function families?
   - How can one guarantee convergence of the training?
   - Why do neural networks generalize so well?
   - How much trainings data is enough?

Geometric Limitations of Current Generative Networks

- Most generative networks use **Euclidean geometry** in latent space
  - Euclidean metric for metric learning
  - Gaussian latent distributions
  - Ordinary gradients for training
- Since networks are continuous mappings, this translates to the data space
  - Cannot exactly model
    - Manifolds with arbitrary topology
    - Data supported on multiple disconnected compact regions
    - Non-Euclidean group symmetries
  - If latent dimension is high enough, accurate approximate embeddings are still possible, but
    - Representation does not have correct intrinsic dimension
    - No disentanglement into meaningful concepts
    - Representation artifacts (e.g. incorrect similarities, outliers not recognized, …)

⇨ **Interpretability requires that latent structure is compatible with intrinsic structure of the data**

Example: Embedding of the WordNet Hierarchy

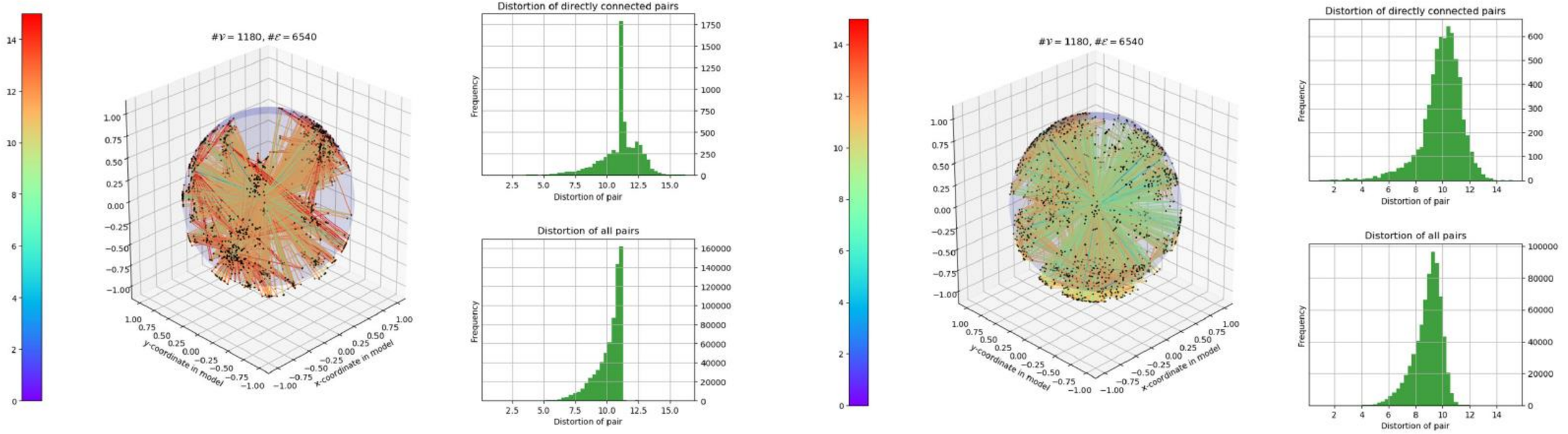
- Famous paper by Nickel and Kiela (2017)
  - Hierarchies have intrinsically hyperbolic structure
  - ⇨ Embeddings in hyperbolic space are much more accurate

			Dimensionality					
			5	10	20	50	100	200
WordNet Reconstruction	**Euclidean**	Rank	3542.3	2286.9	1685.9	1281.7	1187.3	1157.3
		MAP	0.024	0.059	0.087	0.140	0.162	0.168
	Translational	Rank	205.9	179.4	95.3	92.8	92.7	91.0
		MAP	0.517	0.503	0.563	0.566	0.562	0.565
	Poincaré	Rank	4.9	4.02	3.84	3.98	3.9	**3.83**
		MAP	0.823	0.851	0.855	0.86	0.857	**0.87**
WordNet Link Pred.	**Euclidean**	Rank	3311.1	2199.5	952.3	351.4	190.7	81.5
		MAP	0.024	0.059	0.176	0.286	0.428	0.490
	Translational	Rank	65.7	56.6	52.1	47.2	43.2	40.4
		MAP	0.545	0.554	0.554	0.56	0.562	0.559
	Poincaré	Rank	5.7	**4.3**	4.9	4.6	4.6	4.6
		MAP	0.825	0.852	0.861	**0.863**	0.856	0.855

- 5-dimensional hyperbolic embedding more accurate than 200-dimensional Euclidean

Example: Embedding of the WordNet Hierarchy

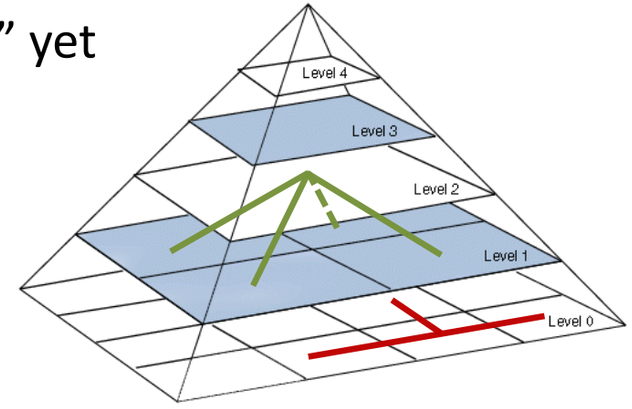
- Famous paper by Nickel and Kiela (2017)
  - Hierarchies have intrinsically hyperbolic structure
  - ⇨ Embeddings in hyperbolic space are much more accurate

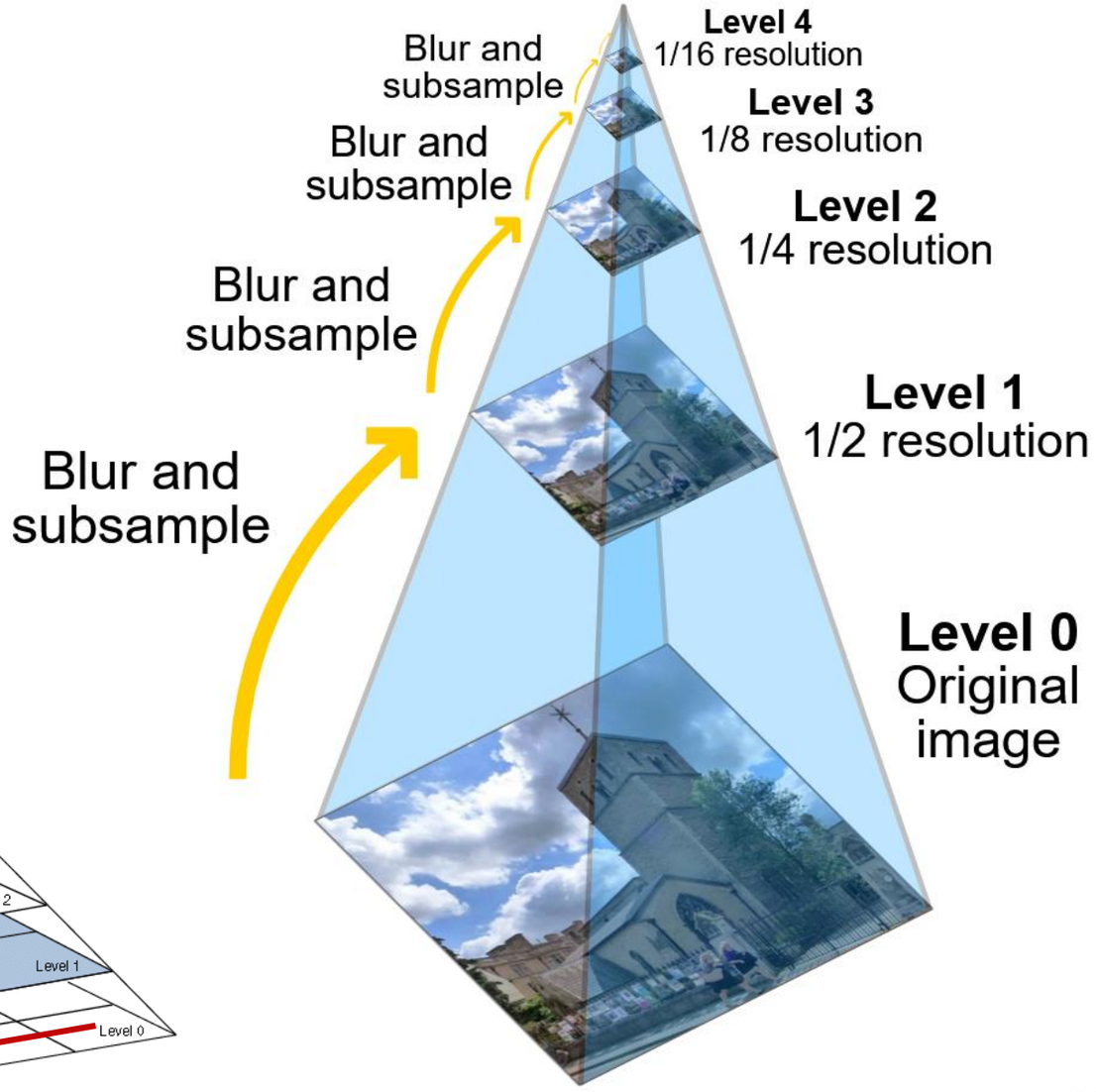


- 5-dimensional hyperbolic embedding more accurate than 200-dimensional Euclidean
- 3-dimensional hyperbolic space suffices with better initialization (Clemens Fruböse)

Example: Image Pyramid

- Classical representation in image analysis
  - Alternate blurring and down-sampling
  - Get gradually coarser and more abstract images
- Two kinds of relations:
  - Spatial: context with in single level
    (e.g. **grid neighbors**) ⇨ Euclidean
  - Scale: objects with their constituents
    (e.g. **pyramid neighbors**) ⇨ Hyperbolic (?)
  - No satisfactory “scale embedding” yet
    allowing, for example latent
    interpolation of size
  - Scale-space theory failed
    Wavelets are not interpretable





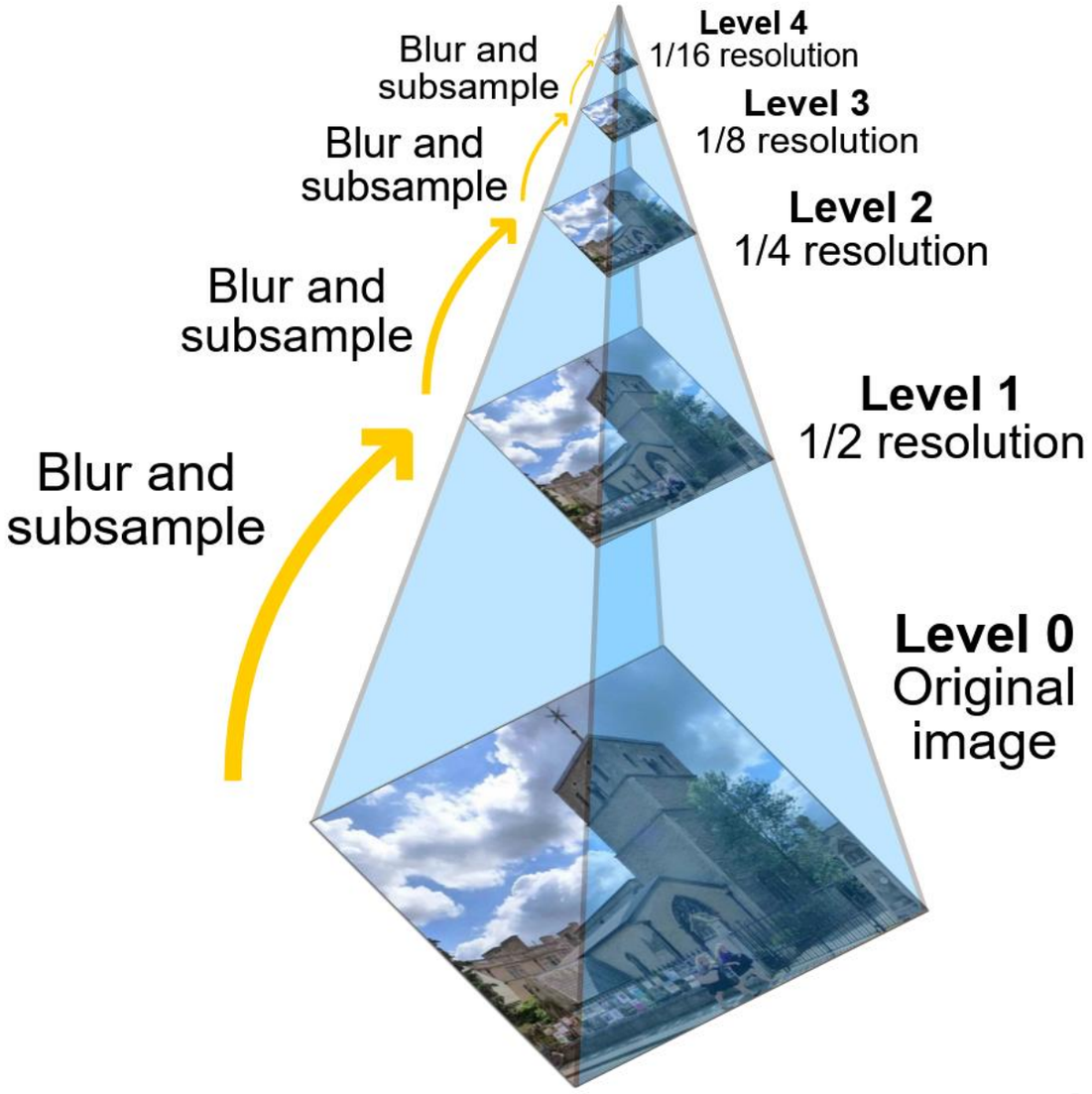
Summary

- Generative models with structured latent spaces are a promising approach to interpretable ML
- Most networks use Euclidean latent geometry
- Results in sub-optimal representations when the geometry/topology of the data differs

⇨ Networks with other latent geometries, group equivariance, Riemannian gradient descent are hot research topics

⇨ I'm especially interested in hyperbolic geometry for "learning to abstract"

⇨ What types of network mappings $\boldsymbol{z} = f_\theta(\boldsymbol{x})$ do we want and how can we get them?



Invertible Neural Networks (INNs) with Coupling Layers

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

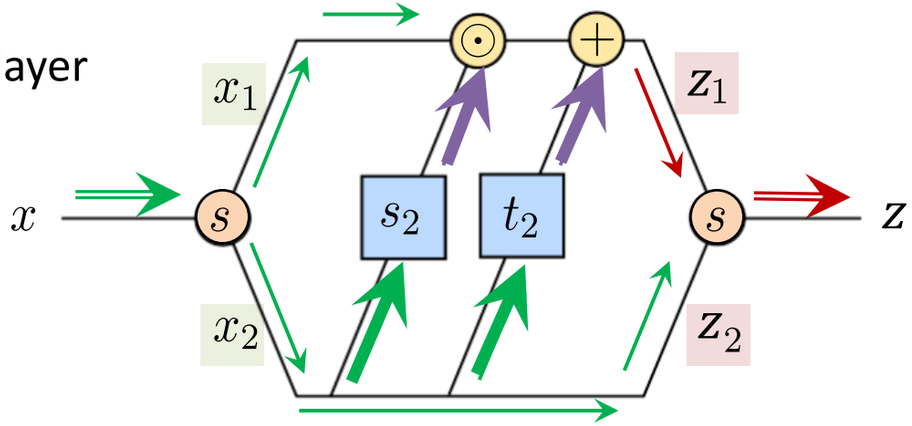
- Network is a sequence of *affine coupling layers*
- Each coupling layer splits its input $x \in \mathbb{R}^D$ into two halves $x_1, x_2 \in \mathbb{R}^{D/2}$
- Upper half is subjected to an affine transformation ⇨ outputs $z_1, z_2 \in \mathbb{R}^{D/2}$
- Affine coefficients are computed by standard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

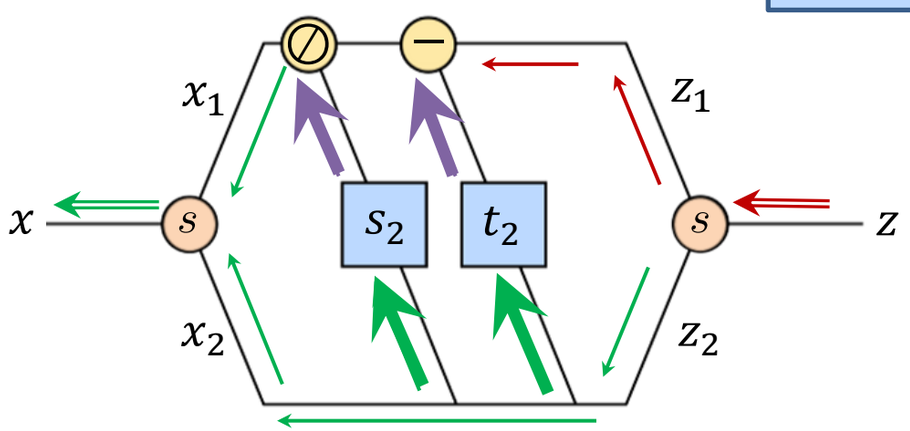
Forward computation: $z_1 = x_1 \odot s_2(x_2) + t_2(x_2), \quad z_2 = x_2$

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

nested networks s_2 and t_2 are always executed in the same direction

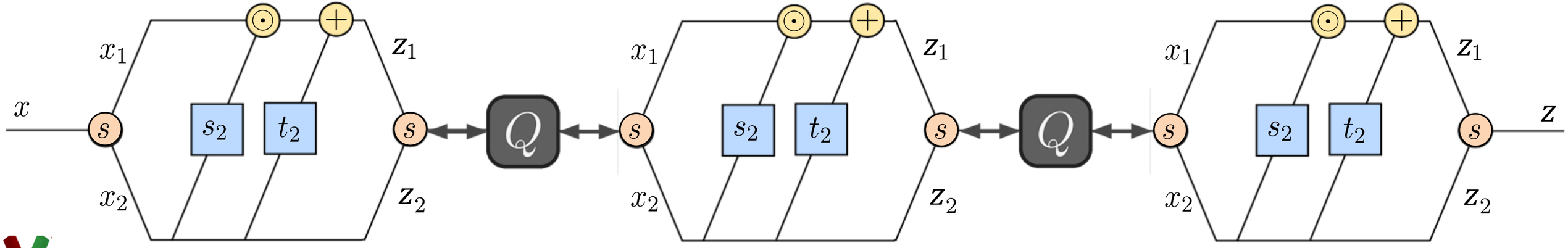
Coupling layer





Deep INNs

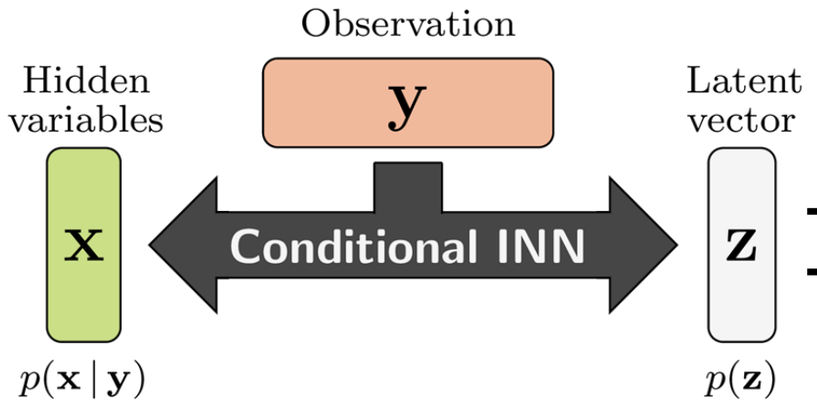
- Concatenate many coupling layers
- Alternate with orthogonal layers Q
  - ⇨ Active (upper lane) and passive (lower lane) dimensions change in each layer
  - – Random permutations or projections are good enough, learning Q is not necessary
- Surprisingly powerful despite its simplicity
- Similar to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



Conditional INNs (cINNs)

Model conditional posterior $p(\boldsymbol{x} \mid \boldsymbol{y})$

- Receive observation y as additional conditioning input



Training

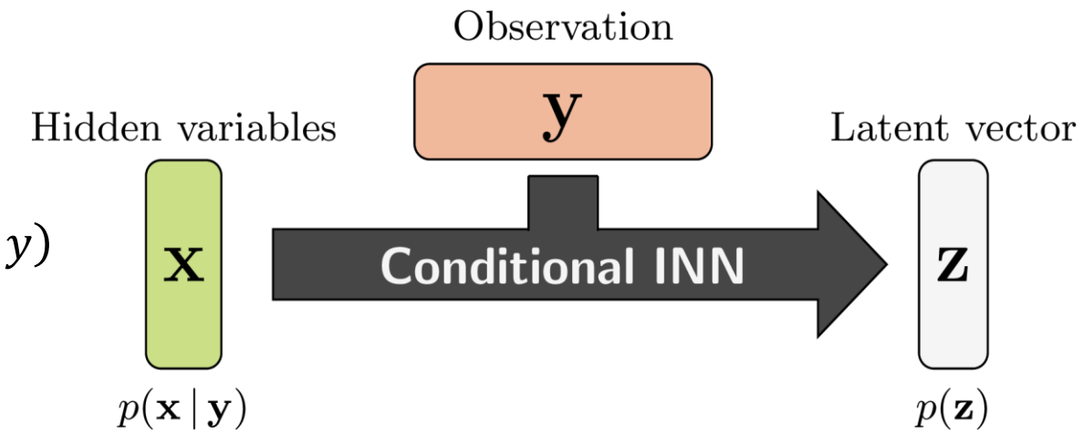
Given training pair (x, y)

run cINN forward:

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

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

(maximum likelihood loss)



Inference

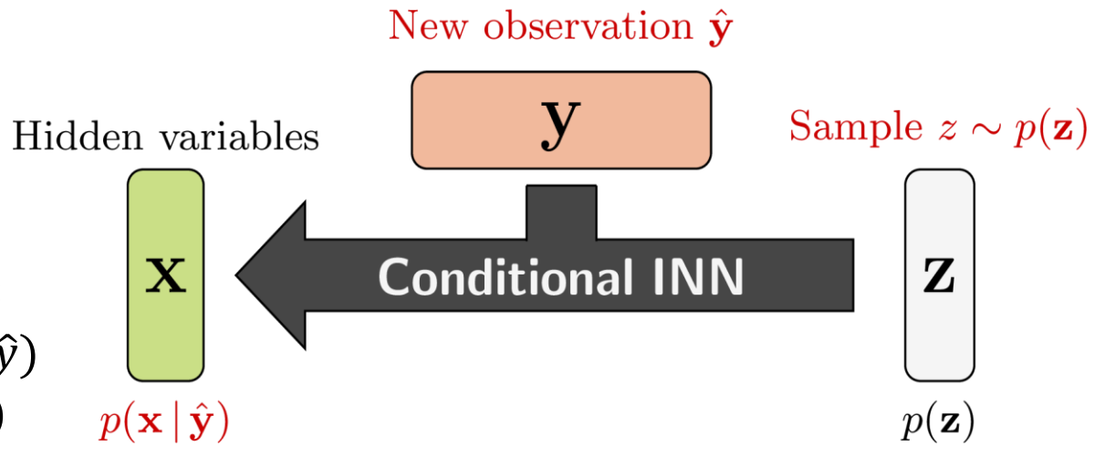
Given observation $\hat{y}$

run cINN backward:

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

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

⇨ $x \sim p(x \mid \hat{y})$



cINN Training

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

- Let the cINN represent the function $z = f_\theta(x\,; y)$
- 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:

$$\hat{\theta} = \arg\max_\theta \sum_{i \in \mathcal{TS}} \hat{p}(x_i \mid y_i)$$

$$= \arg\min_\theta \sum_{i \in \mathcal{TS}} \left(\frac{\|f_\theta(x_i; y_i)\|_2^2}{2} - \sum_{l=1}^{L} \log(\|s_l(x_i; y_i\|_1) \right)$$

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

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

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

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

We can do much better using invertible neural networks!

  - 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

Multiple Possibilities for INNs

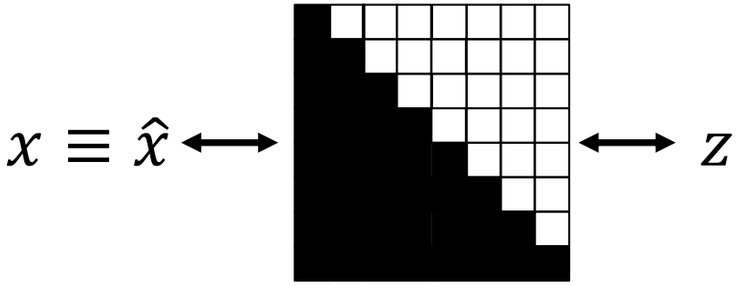
Autoregressive Models

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.



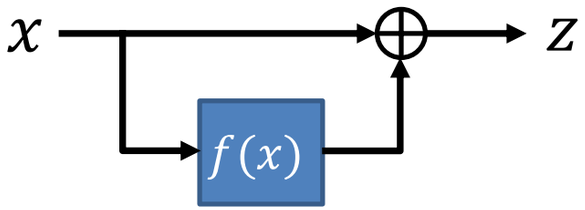
inverse direction inefficient

⇨ use two complementary nets

example: parallel WaveNet

iResNets
(invertible residual networks)

Residual block:



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

is invertible when

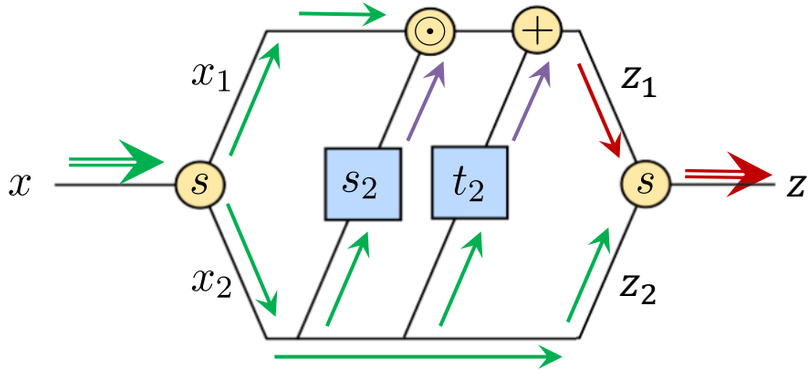
$$\|f(x)\|_{\text{Lipshitz}} < 1$$

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