Wrocław University of Science and Technology

Department of Artificial Intelligence

genwro.AI

Normalizing Flows - fundamental concepts and applications in counterfactual explanations

Maciej Zięba

GEIST Research Group 18.03.2025

Wrocław University of Science and Technology

Department of Artificial Intelligence

Department of Artificial Intelligence @ Wroclaw Tech

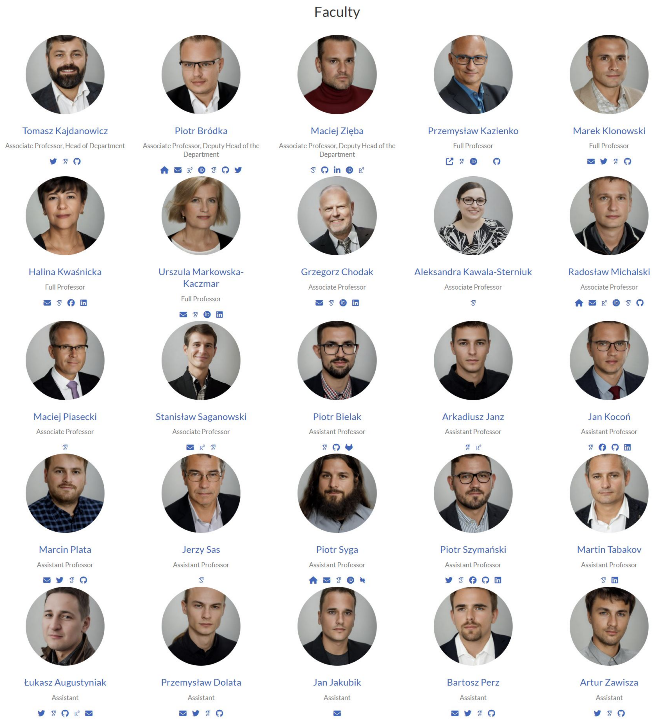
Department of AI

- 5 full professors
- 8 associate professors
- 13 assistant professors
- over 50 PhD students

Master studies in AI 60 students per year

Just starting Engineering studies

https://ai.pwr.edu.pl/



genwro.ai Research Group

Research areas

- Generative modelling
- Uncertainty models
- Counterfactual representations
- 3D representations
- Few-shot learning
- Image enhancement

https://genwro.ai.pwr.edu.pl

Generative models - modern applications

Conditional image generation

seed

1338

steps

20

width

512

height

728

prompt

RAW photo, a portrait photo of a latino man in casual clothes, natural skin, 8k uhd, high quality, film grain, Fujifilm XT3

guidance

5

scheduler

EulerA

negative_prompt

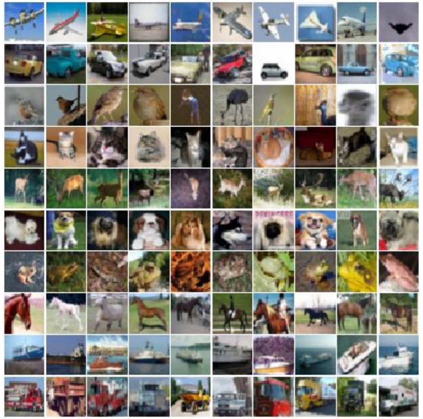
(deformed iris, deformed pupils, semi-realistic, cgi, 3d, render, sketch, cartoon, drawing, anime:1.4), text, close up, cropped, out of frame, worst quality, low quality, jpeg artifacts, ugly, duplicate, morbid, mutilated, extra fingers, mutated hands, poorly drawn hands, poorly drawn face, mutation, deformed, blurry, dehydrated, bad anatomy, bad proportions, extra limbs, cloned face, disfigured, gross proportions, malformed limbs, missing arms, missing legs, extra arms, extra legs, fused fingers, too many fingers, long neck

Generative models - preliminaries

Our goal is to find some approximation of true data distribution

$$p(\mathbf{x})$$

But we have access only to the data examples



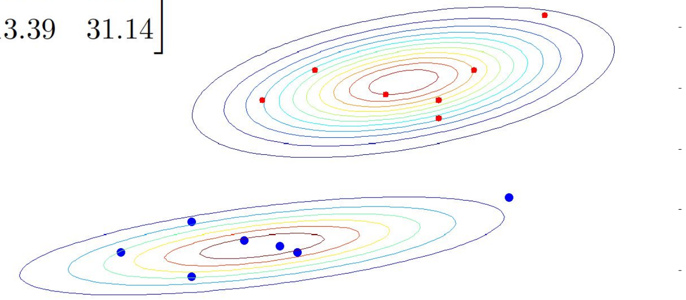
Generative models - standard approach

Standard approach assumes:

- *Select some **well known distribution** as true data approximation.*
- *Get the parameters by **ML/MAP estimation.***
- *Sample examples from **approximation**.*

$$\boldsymbol{\mu}_1 = [184.29,\ 91.14]$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 29.57 & 13.39 \\ 13.39 & 31.14 \end{bmatrix}$$



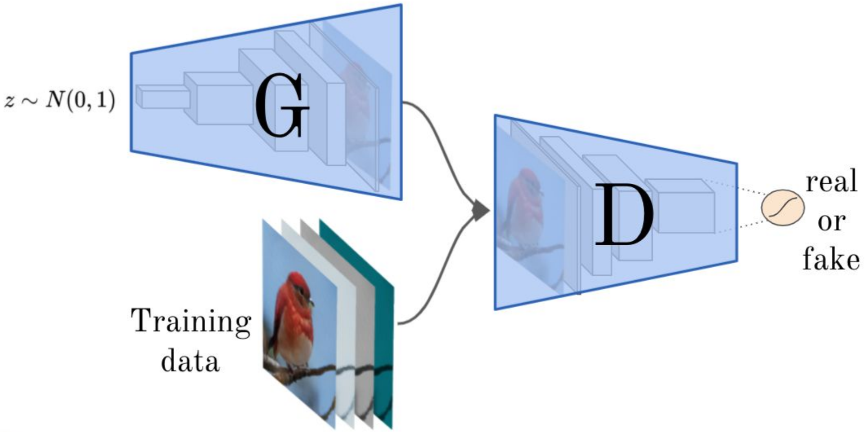
$$\boldsymbol{\mu}_0 = [176.00,\ 64.86]$$

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 49.67 & 17.29 \\ 17.29 & 17.13 \end{bmatrix}$$

Generative models - GANs

Our goal is to find some approximation of true data distribution

$$p(\mathbf{x})$$



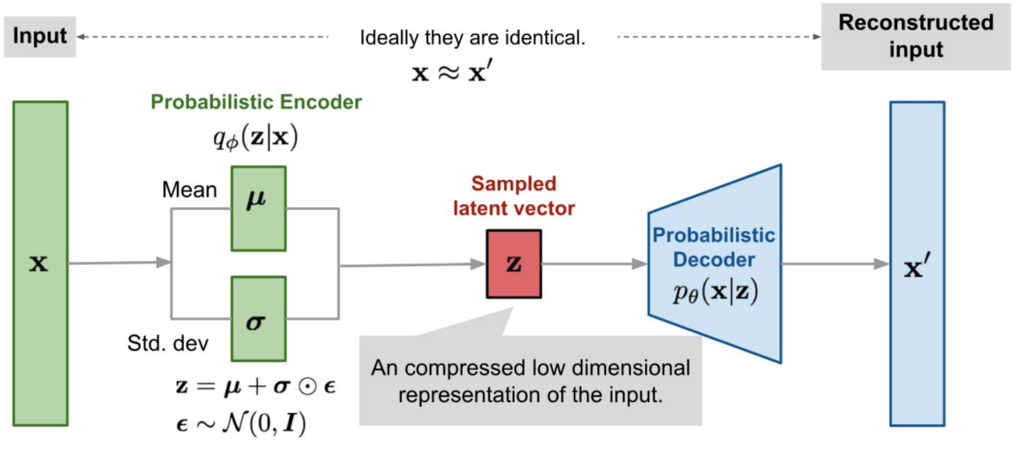
we can try to sample from p(x) without knowing the explicit form - GAN is some solution

Generative models - VAEs

Our goal is to find some approximation of true data distribution

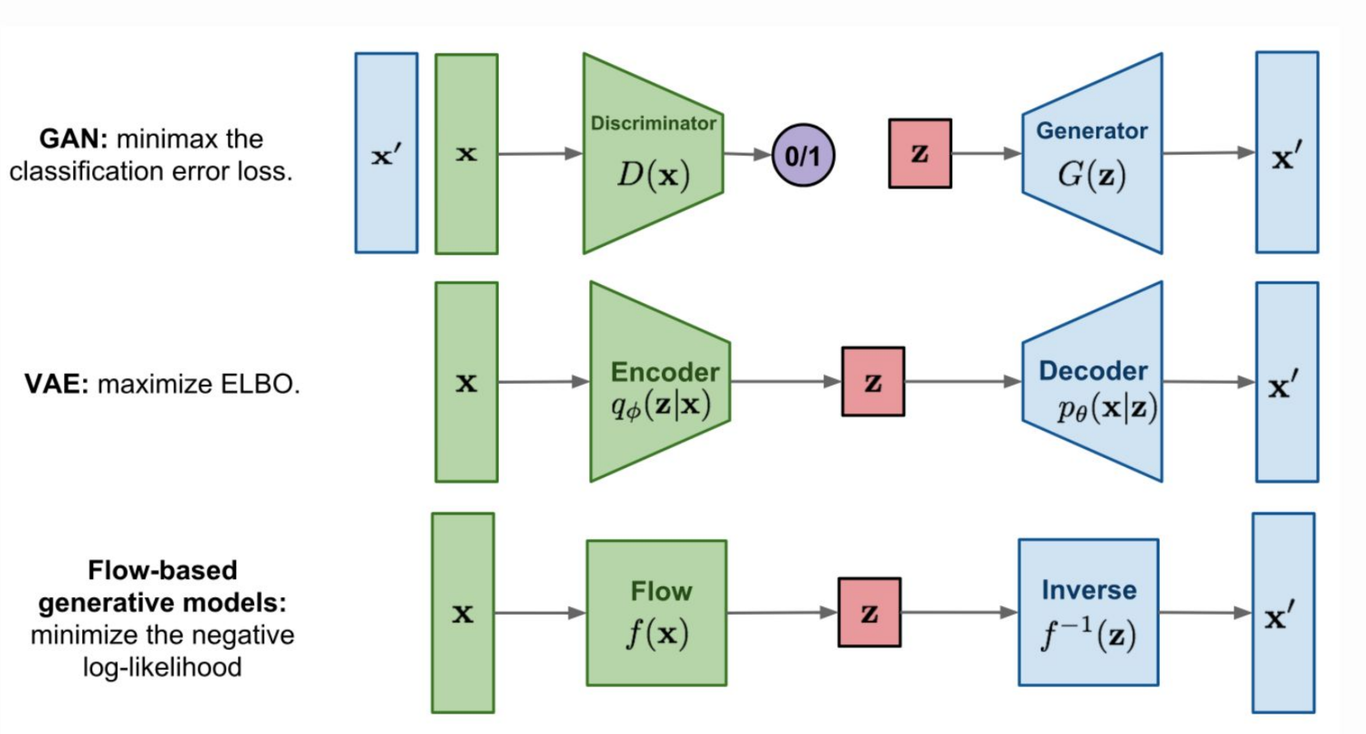
$$\ln p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left[\ln \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]$$

We can estimate lower bound for p(x)



Source: https://lilianweng.github.io/

Generative models - normalizing flows



Source: https://lilianweng.github.io/

Normalizing Flows - basic concepts

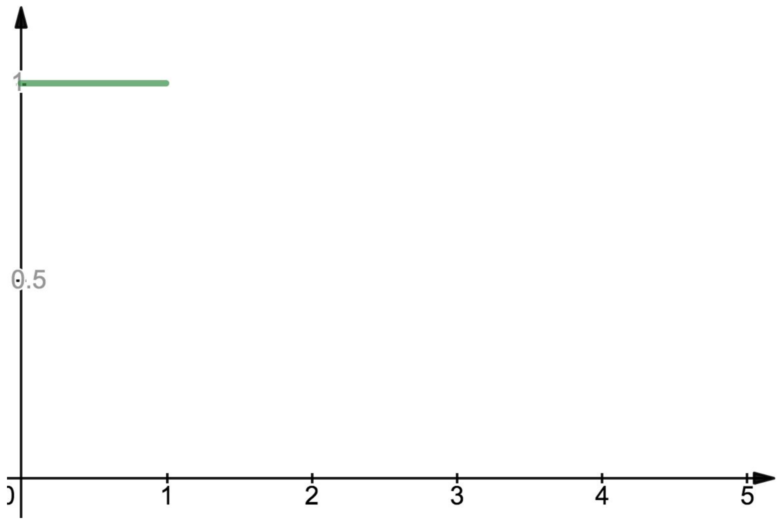
Change of variable formula - example

Consider density function for uniform distribution:

$$p_X(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & otherwise \end{cases}$$

We create a new random variable using the following transformation:

$$Y = f(X) = \sqrt{X}$$

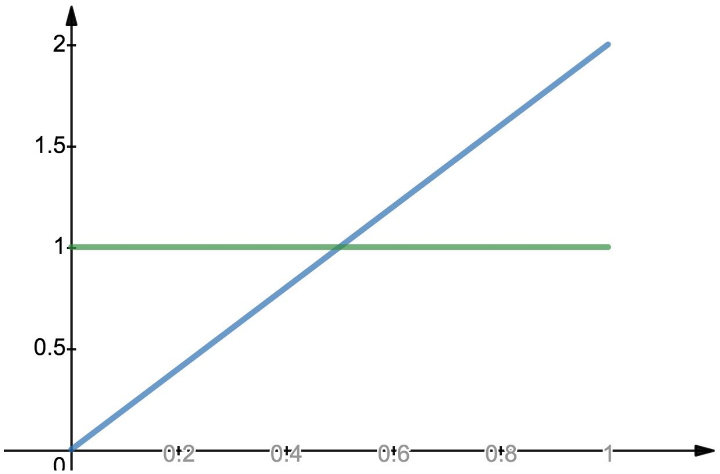


What is density function for a new variable Y?

Change of variable formula - example

The new density function can be defined as:

$$p(y) = \begin{cases} 2y & 0 <= y <= 1 \\ 0 & otherwise \end{cases}$$



Thanks to change of variable formula:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{df^{-1}(y)}{dy} \right|$$

where:

$$f^{-1}(Y) = Y^2$$

Change of variable formula - multidimensional case

for multidimensional case:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{df^{-1}(y)}{dy} \right|$$

becomes:

$$p_{\mathbf{Y}}(\mathbf{y}) = p_{\mathbf{X}}(\mathbf{f}^{-1}(\mathbf{y}))|\det \mathbf{J}_{\mathbf{f}^{-1}}|$$

where:

$$\mathbf{J}_{\mathbf{f}^{-1}} = \begin{bmatrix} \frac{\partial f_1^{-1}}{\partial y_1} & \cdots & \frac{\partial f_1^{-1}}{\partial y_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D^{-1}}{\partial y_1} & \cdots & \frac{\partial f_D^{-1}}{\partial y_D} \end{bmatrix}$$

Change of variable formula - multidimensional case

It also works for:

$$p_X(x) = p_Y(f(x))) \left| \frac{df(x)}{dx} \right|$$

where:

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x})) \left| \det \mathbf{J_f} \right|$$

and:

$$\mathbf{J_f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial x_1} & \cdots & \frac{\partial f_D}{\partial x_D} \end{bmatrix}$$

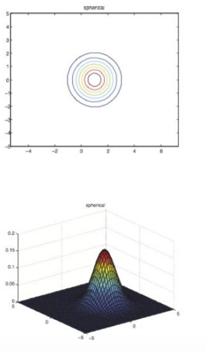
Change of variable formula for normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\,|\det \mathbf{J}_{\mathbf{f}}|$$

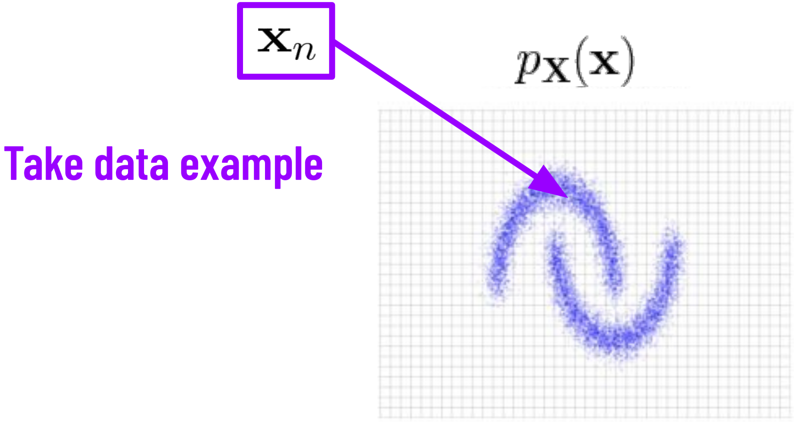
data distribution

base distribution with
simple density function

Invertible mapping
(parameterized)



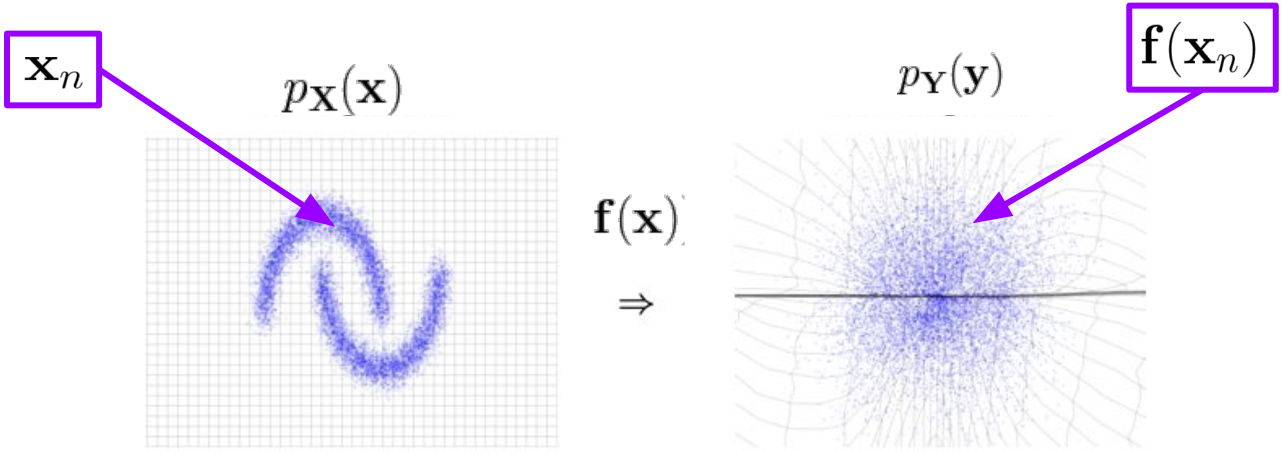
Inference with normalizing flows



Source: https://arxiv.org/abs/1605.08803

Inference with normalizing flows

Transform it to Y space with known base distribution

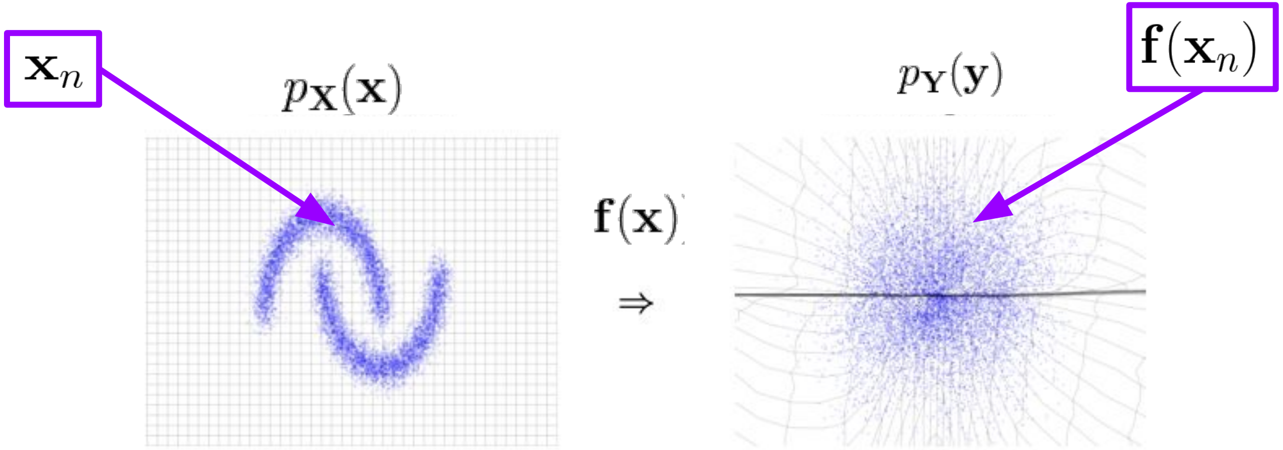


Source: https://arxiv.org/abs/1605.08803

Inference with normalizing flows

$$p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}_n))$$

Get the density value in Y space

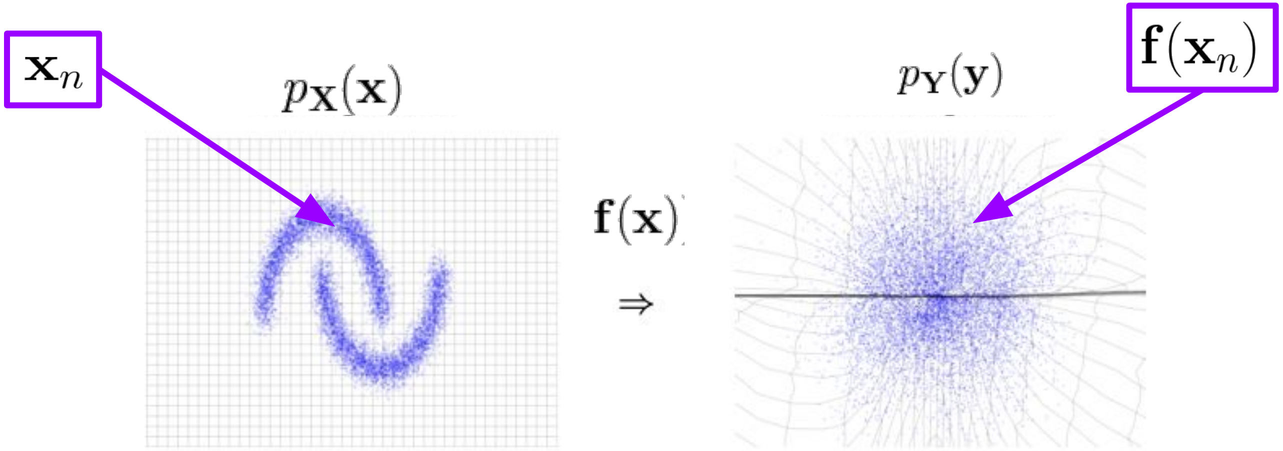


Source: https://arxiv.org/abs/1605.08803

Inference with normalizing flows

$$p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}_n))\ |\det \mathbf{J_f}|$$

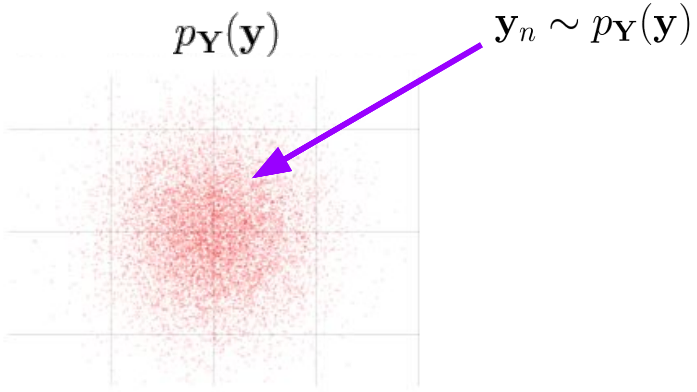
Scale by determinant of Jacobian



Source: https://arxiv.org/abs/1605.08803

Sampling with normalizing flows

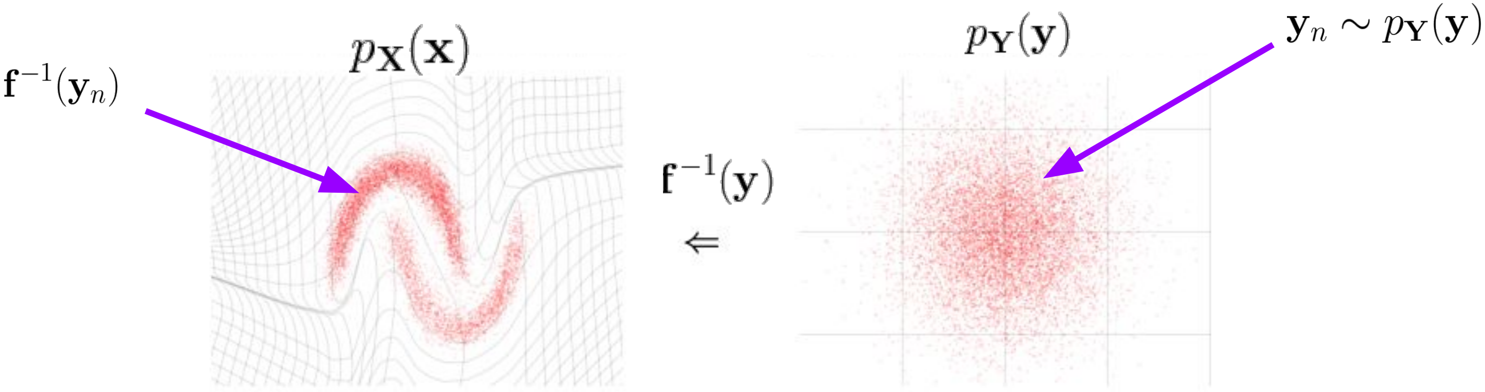
Sample from known base distribution



Source: https://arxiv.org/abs/1605.08803

Sampling with normalizing flows

Apply invert transform to obtain sample from data distribution



Source: https://arxiv.org/abs/1605.08803

Training normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\, |\det \mathbf{J}_{\mathbf{f}}|$$

We assume that invertible transformation is parametrized:

$$\mathbf{f}_{\boldsymbol{\theta}} := \mathbf{f}$$

Training normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\,|\det \mathbf{J}_{\mathbf{f}}|$$

We assume that invertible transformation is parametrized:

$$\mathbf{f}_{\boldsymbol{\theta}} := \mathbf{f}$$

We have access to training data:

$$\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$$

Training normalizing flows

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\left|\det \mathbf{J}_{\mathbf{f}}\right|$$

We assume that invertible transformation is parametrized:

$$\mathbf{f}_{\boldsymbol{\theta}} := \mathbf{f}$$

We have access to training data:

$$\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$$

We optimize negative log-likelihood to obtain the best parameters:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} - \sum_{n=1}^{N} \log(p_{\mathbf{X}(\mathbf{x}_n)})$$

Normalizing flows - challenges

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\left|\det \mathbf{J}_{\mathbf{f}}\right|$$

The choice of invertible function

Normalizing flows - challenges

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Y}}(\mathbf{f}(\mathbf{x}))\,\left|\det \mathbf{J}_{\mathbf{f}}\right|$$

The choice of invertible function

Determinant of Jacobian is difficult to calculate for high-dimensional data

Discrete normalizing flows - NICE

Make use of so called coupling layers - sequence of the following operations:

$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 + m(x_1) \end{cases}$$

Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).

Discrete normalizing flows - NICE

Make use of so called coupling layers - sequence of the following operations:

$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 + m(x_1) \end{cases}$$

m() is neural network

Not need to be invertible!

Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).

Discrete normalizing flows - NICE

Invert is easy to calculate:

$$\begin{cases} x_1 = y_1 \\ x_2 = y_2 - m(x_1) \end{cases}$$

Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).

Discrete normalizing flows - NICE

and determinant of Jacobian is easy to calculate:

$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 + m(x_1) \end{cases}$$

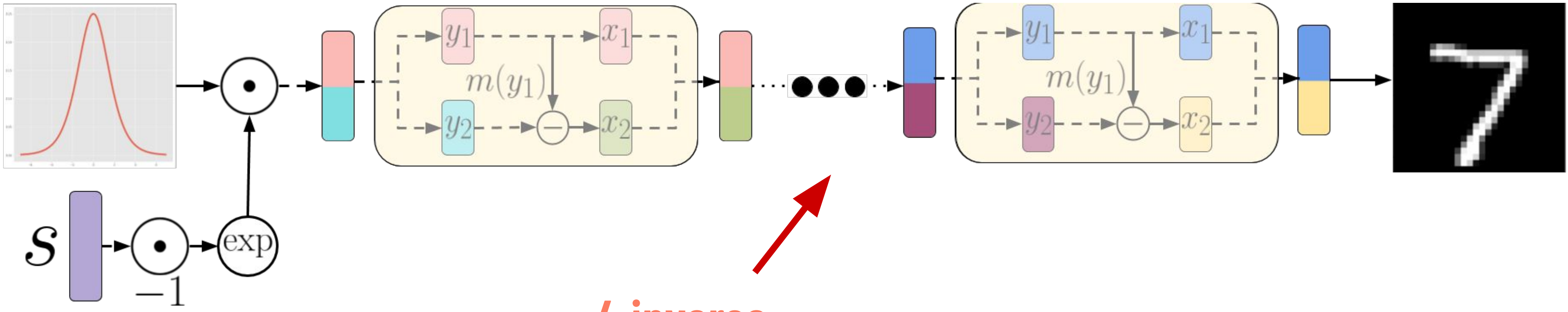
$$\mathbf{J_f} = \frac{\partial y}{\partial x} = \begin{bmatrix} I & 0 \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix}$$

$$\det \frac{\partial y_2}{\partial x_2} = 1 \Rightarrow \det \frac{\partial y}{\partial x} = 1$$

Dinh, Laurent, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516* (2014).

Coupling Layers working together

Inverse transformation



4 inverse coupling layers

scaling parameter

Discrete normalizing flows - RealNVP

$$y_{1:d} = x_{1:d}$$
$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$

s() and t() are neural networks
Not need to be invertible!

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." *arXiv preprint arXiv:1605.08803* (2016).

Conditional normalizing flows

RealNVP

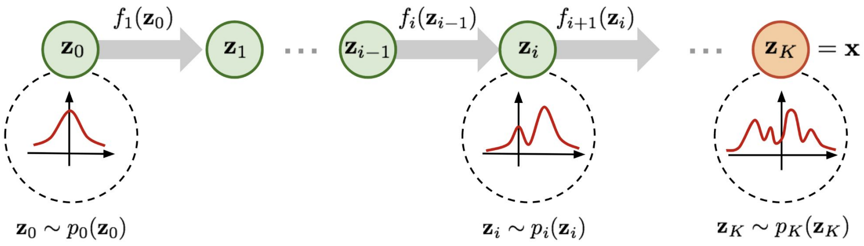
$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 \odot \exp\left(s(x_1)\right) + m(x_1) \end{cases}$$

Conditional RealNVP

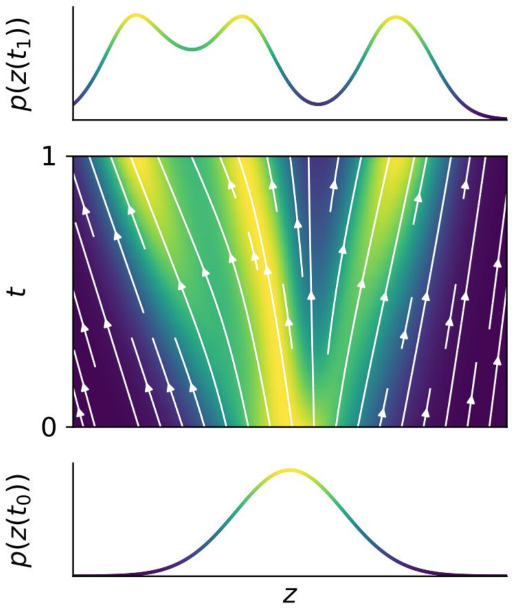
$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 \odot \exp\boxed{\left(s(x_1, z)\right)} + \boxed{m(x_1, z)} \end{cases}$$

Continuous Normalizing flows

Discrete Normalizing Flows



Continuous Normalizing Flows



Applications for counterfactual explanations

Counterfactual explanations with flows

Factual

Age	Income	Debt	Accounts
28	800	200	5

C() = no loan

Counterfactual

Age	Income	Debt	Accounts
28	1000	200	3

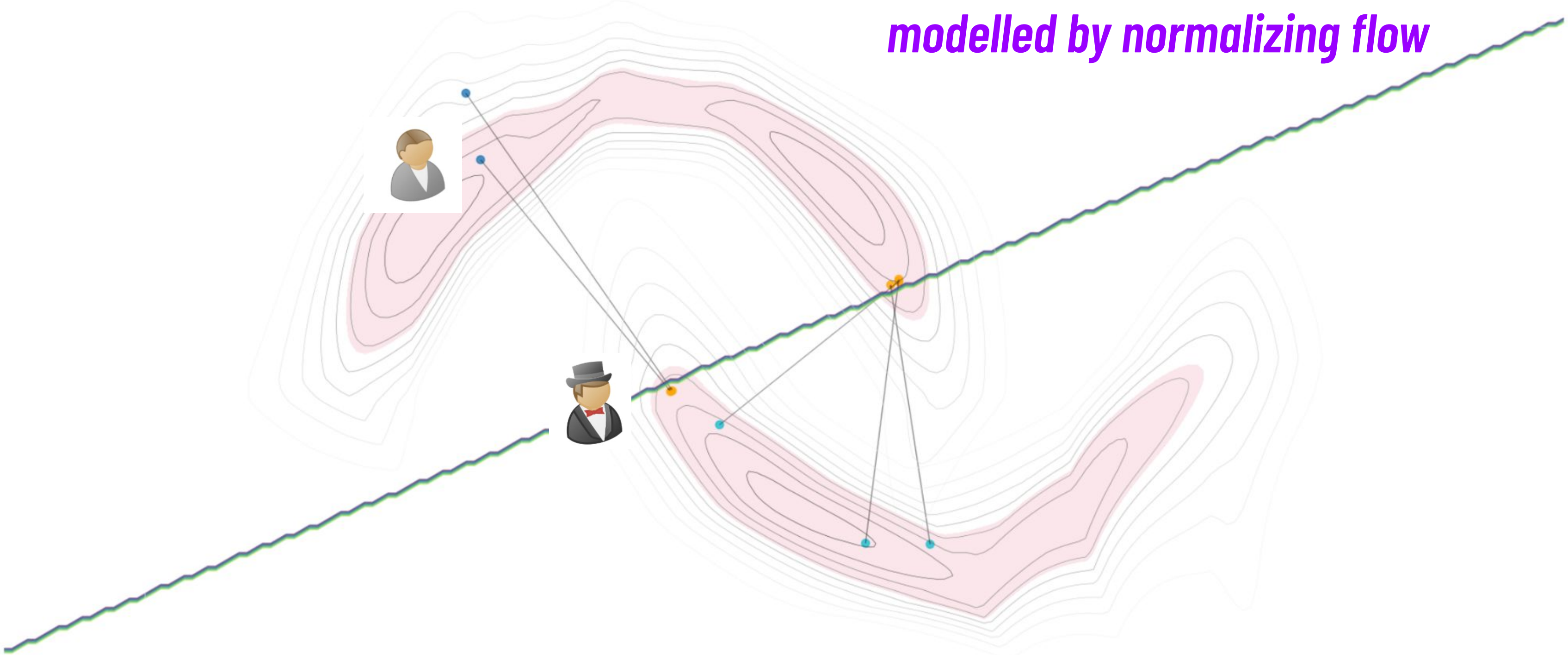
C() = loan

Actionable Recourse:
Your loan will be approved, if you increase income by $200 and close two accounts.

Source: https://relational.ai/resources/machine-learning-in-consumer-credit

Counterfactual explanations with flows

Distribution of plausible regions modelled by normalizing flow



Source: Wielopolski P, Furman O, Stefanowski J, Zięba M. Probabilistically Plausible Counterfactual Explanations with Normalizing Flows. ECAI 2024

Counterfactual explanations with flows

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} d(\mathbf{x}_0, \mathbf{x}') + \lambda \cdot \left(\ell_v(\mathbf{x}', y') + \ell_p(\mathbf{x}', y') \right)$$

distance between original example and counterfactual

Source: Wielopolski P, Furman O, Stefanowski J, Zięba M. Probabilistically Plausible Counterfactual Explanations with Normalizing Flows. ECAI 2024

Counterfactual explanations with flows

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} d(\mathbf{x}_0, \mathbf{x}') + \lambda \cdot \Big(\ell_v(\mathbf{x}', y') + \ell_p(\mathbf{x}', y') \Big)$$

validity loss to guarantee correct classification

$$\ell_v(\mathbf{x}', y') = \max\Big(0.5 + \epsilon - p_d(y'|\mathbf{x}'), 0\Big)$$

Source: Wielopolski P, Furman O, Stefanowski J, Zięba M. Probabilistically Plausible Counterfactual Explanations with Normalizing Flows. ECAI 2024

Counterfactual explanations with flows

$$\arg\min_{\mathbf{x}' \in \mathbb{R}^d} d(\mathbf{x}_0, \mathbf{x}') + \lambda \cdot \left(\ell_v(\mathbf{x}', y') + \ell_p(\mathbf{x}', y') \right)$$

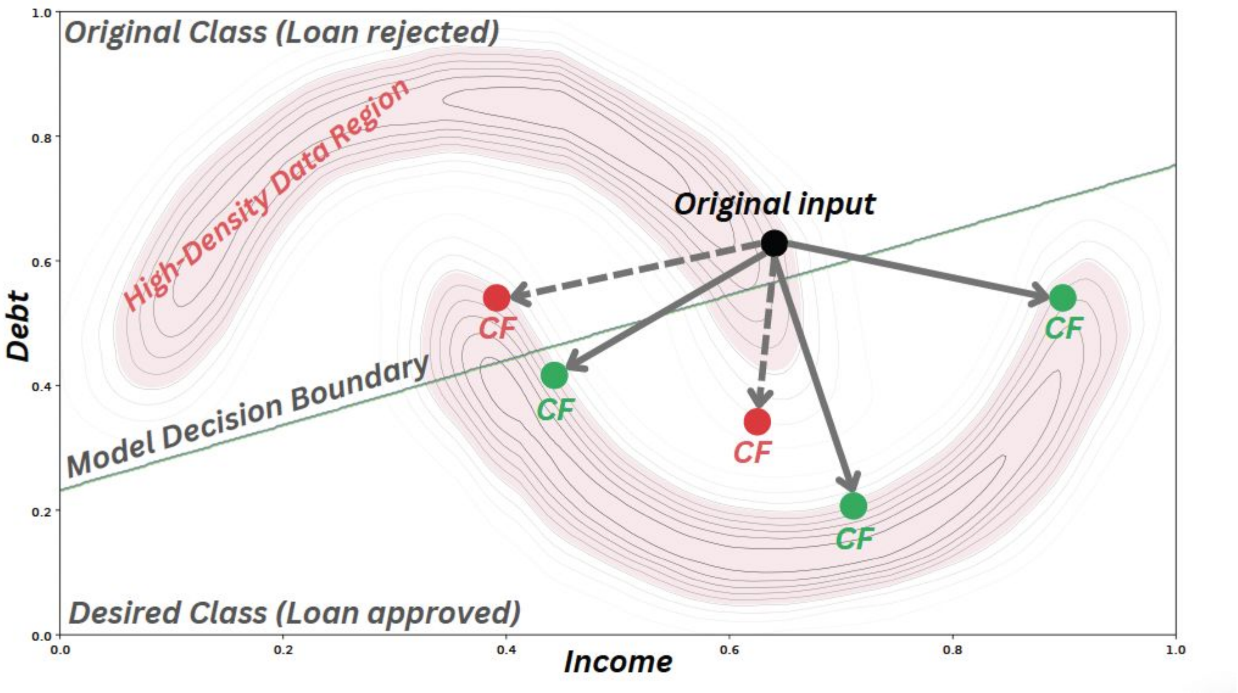
plausibility to get
in-distribution
sample

$$\ell_p(\mathbf{x}', y') = \max\left(\delta - p(\mathbf{x}' | y'), 0 \right)$$

modelled with normalizing flow

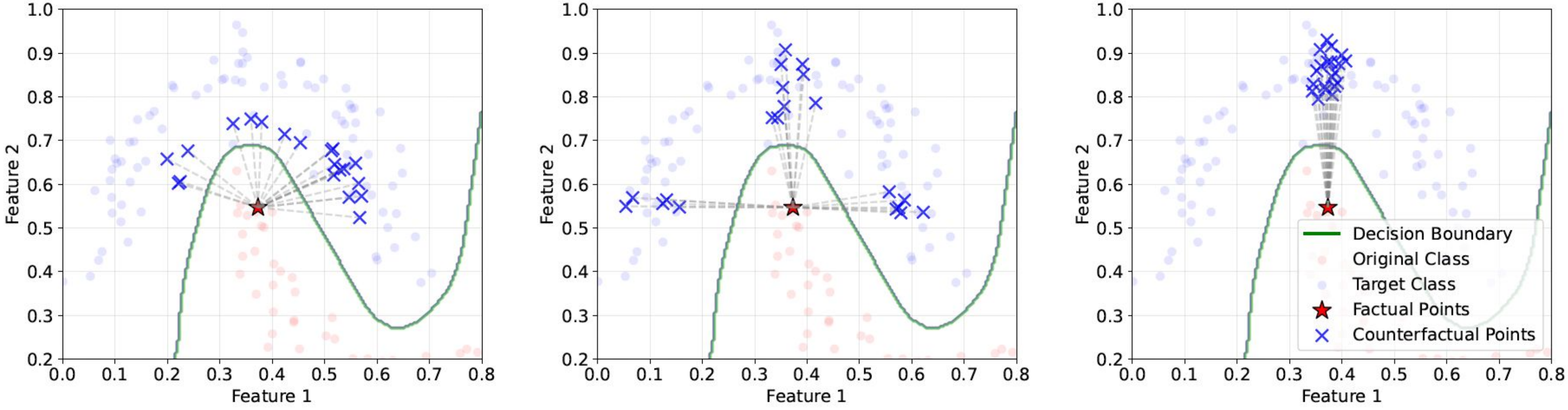
Source: Wielopolski P, Furman O, Stefanowski J, Zięba M. Probabilistically Plausible Counterfactual Explanations with Normalizing Flows. ECAI 2024

Generating counterfactual with flows



Source: Furman, Oleksii, Ulvi Movsum-zada, Patryk Marszalek, Maciej Zięba, and Marek Śmieja. "DiCoFlex: Model-agnostic diverse counterfactuals with flexible control." NeurIPS 2025.

Generating counterfactual with flows



(a) Unconstrained (b) Sparsity constraints (c) Actionability constraints

Source: Furman, Oleksii, Ulvi Movsum-zada, Patryk Marszalek, Maciej Zięba, and Marek Śmieja. "DiCoFlex: Model-agnostic diverse counterfactuals with flexible control." NeurIPS 2025.

Generating counterfactual with flows

Training Objective

Minimize KL divergence between flow p_θ and empirical distribution $\hat{q}$:

$$\mathcal{Q} = -\mathbb{E}_{\mathbf{x},y'}\mathbb{E}_{\mathbf{x}'\sim\hat{q}(\mathbf{x}'|\mathbf{x},y',d_{p,\mathbf{m}})}\left[\log p_\theta(\mathbf{x}'|\mathbf{x},y',p,\mathbf{m})\right]$$

Empirical Distribution $\hat{q}$

Sample K neighbors from target class y':

$$\hat{q}(\mathbf{x}'|\mathbf{x},y',d_{p,\mathbf{m}}) = \begin{cases} \frac{1}{K} & \text{if } \mathbf{x}' \in \mathcal{N}(\mathbf{x},y',d_{p,\mathbf{m}},K) \\ 0 & \text{otherwise} \end{cases}$$

This ensures **validity**, **proximity**, and **plausibility** by construction!

Source: Furman, Oleksii, Ulvi Movsum-zada, Patryk Marszalek, Maciej Zięba, and Marek Śmieja. "DiCoFlex: Model-agnostic diverse counterfactuals with flexible control." NeurIPS 2025.

Generating counterfactual with flows

Sparsity via L_p norm

$$d_{p,\mathbf{m}}(\mathbf{x}, \mathbf{x}') = \alpha \sum_{j=1}^{D} m_j |x_j - x'_j|^p + \sum_{j=1}^{D} (1 - m_j)|x_j - x'_j|^p$$

Actionability via feature masks $\mathbf{m}$

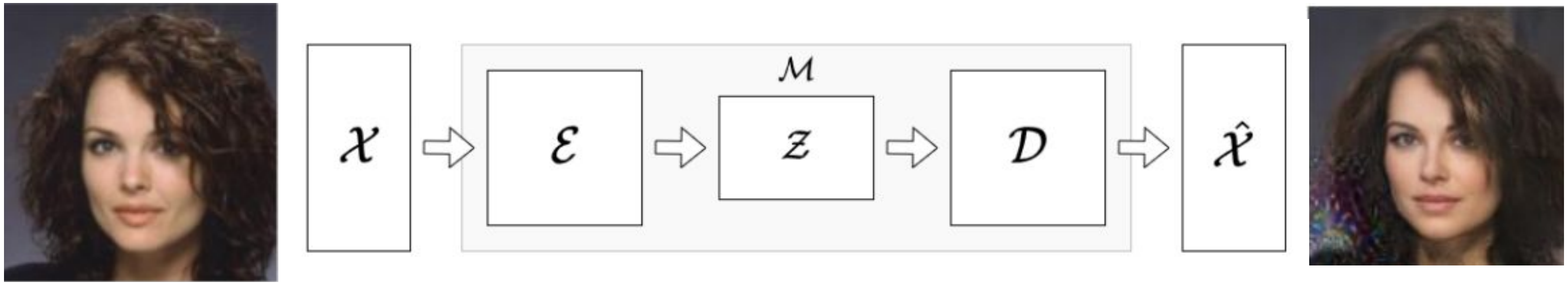
Adjust p and $\mathbf{m}$ at inference — no retraining needed!

Source: Furman, Oleksii, Ulvi Movsum-zada, Patryk Marszalek, Maciej Zięba, and Marek Śmieja. "DiCoFlex: Model-agnostic diverse counterfactuals with flexible control." NeurIPS 2025.

Other applications for normalizing flows

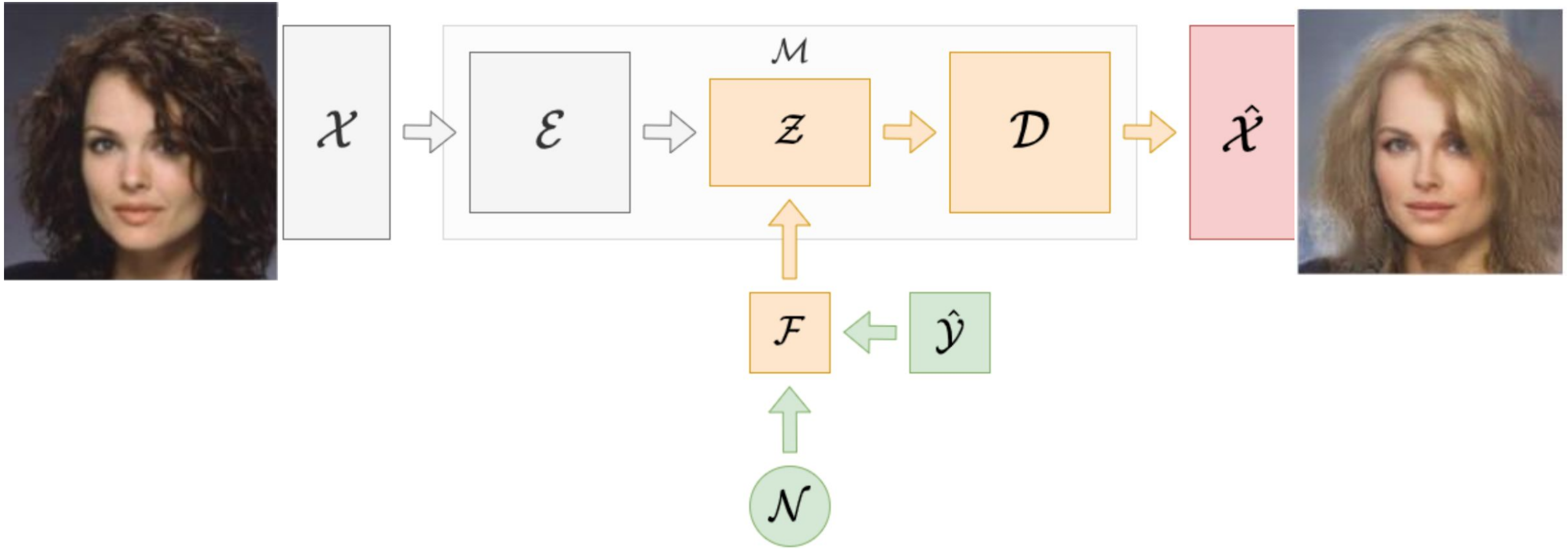
Normalizing flow as a plug-in model for attribute manipulation

We consider trained autoencoder



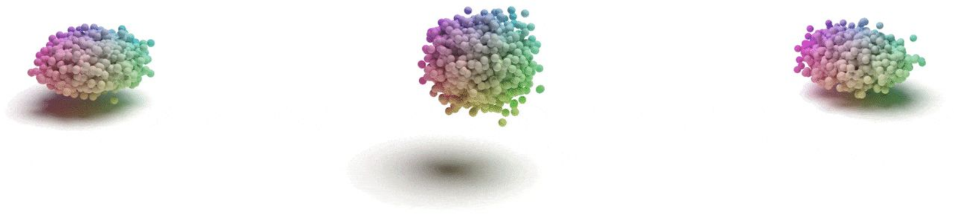
Source: Wielopolski, Patryk, Michał Koperski, and Maciej Zięba. "Flow Plugin Network for conditional generation." arXiv preprint arXiv:2110.04081 (2021).

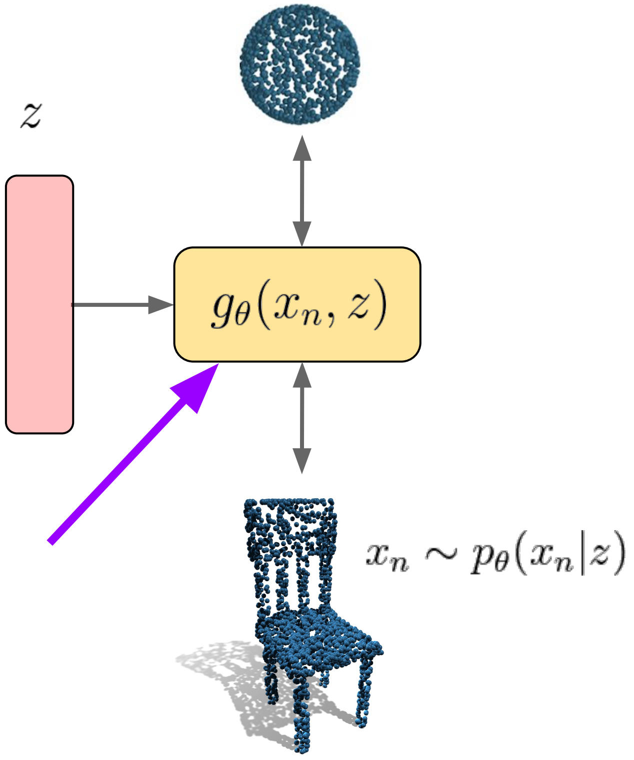
Normalizing flow as a plug-in model for attribute manipulation



Source: Wielopolski, Patryk, Michał Koperski, and Maciej Zięba. "Flow Plugin Network for conditional generation." arXiv preprint arXiv:2110.04081 (2021).

Point cloud generation





Conditional normalizing flow

Source: https://github.com/stevenygd/PointFlow

Probabilistic regression with flows

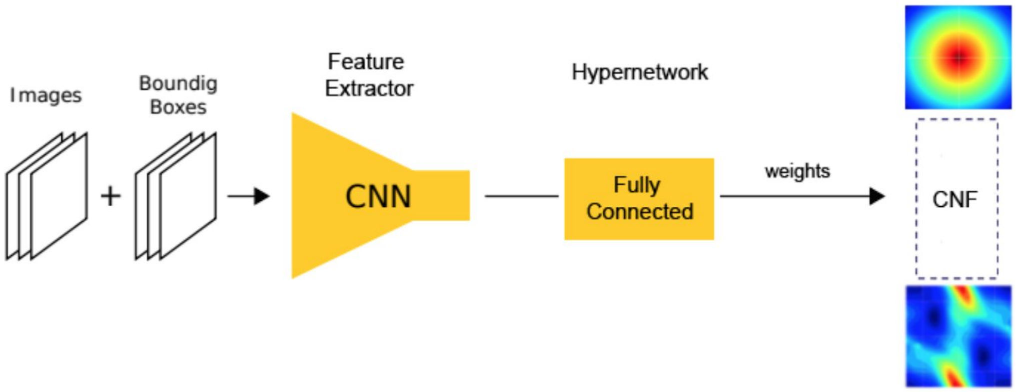
GOAL

Predict location distribution after given period of time

The problem of probabilistic regression modelling

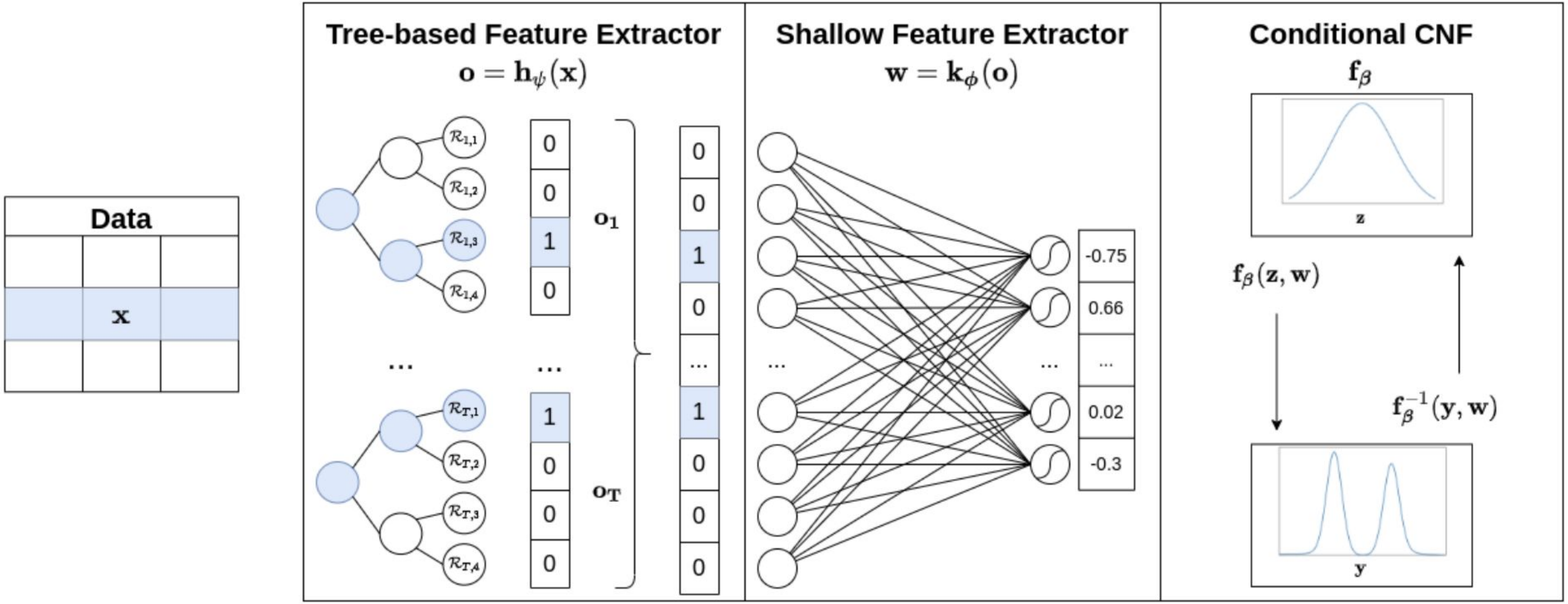
Source: Zięba, Maciej, et al. "RegFlow: Probabilistic Flow-based Regression for Future Prediction." ACIIDS 2024.

Probabilistic regression with flows



Source: Zięba, Maciej, et al. "RegFlow: Probabilistic Flow-based Regression for Future Prediction." ACIIDS 2024.

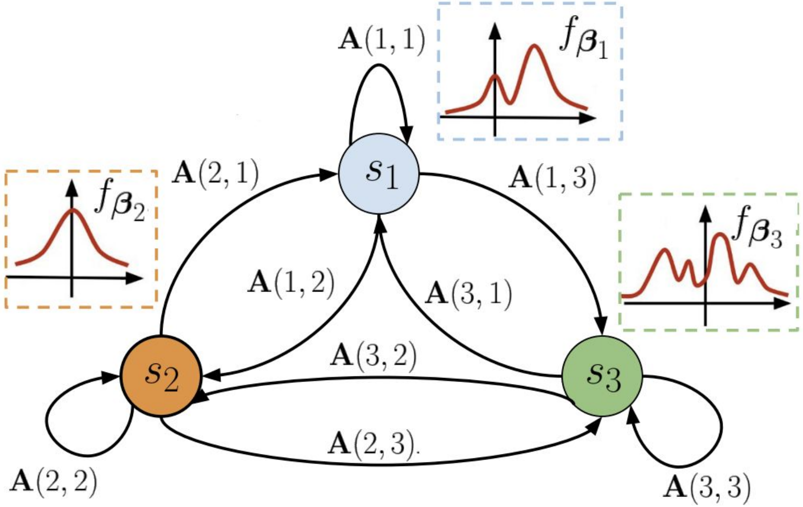
Probabilistic regression with flows



Source: Wielopolski, Patryk, and Maciej Zięba. "TreeFlow: Going beyond Tree-based Gaussian Probabilistic Regression." ECAI 2023

Other flow applications

Figure 1: The concept of FlowHMM for $L = 3$ states and transition matrix $\mathbf{A}$. Each emission distribution characterised by density $f_{\boldsymbol{\beta}_l}(\cdot)$ is modeled using a separate flow component. Thanks to this, they can adjust to complex, non-Gaussian distributions.



Source: Lorek, Pawel, et al. "FlowHMM: Flow-based continuous hidden Markov models." NeurIPS 2022.

Thank you for your attention !!!