Advance course on DL & Geophysical Dynamics

Introduction to Deep Generative models

Patrick Gallinari

Sorbonne Université & CRITEO AI Lab – Paris

patrick.gallinari@sorbonne-universite.fr, p.gallinari@criteo.com

criteo AI Lab

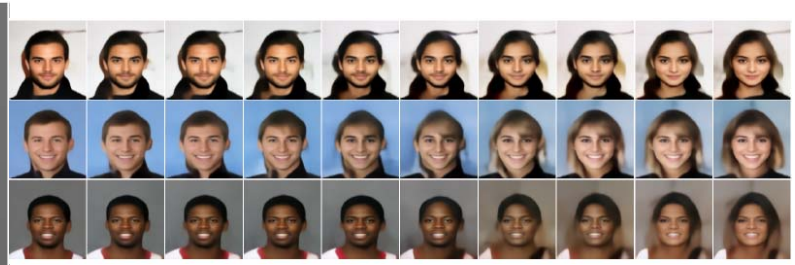
SORBONNE UNIVERSITÉ

Generative models

- Objective
  - Learn a probability distribution model from data samples
    - Given $x^1, \ldots, x^N \in R^n$ learn to approximate their underlying distribution $\mathcal{X}$
    - For complex distributions, there is no analytical form, and for large size spaces (R^n) approximate methods (e.g. MCMC) might fail
    - Deep generative models recently attacked this problem with the objective of handling large dimensions and complex distributions

https://en.wikipedia.org/wiki/Edmond_de_Belamy
432 k$ Christies in 2018

Xie et al. 2019
artificial smoke



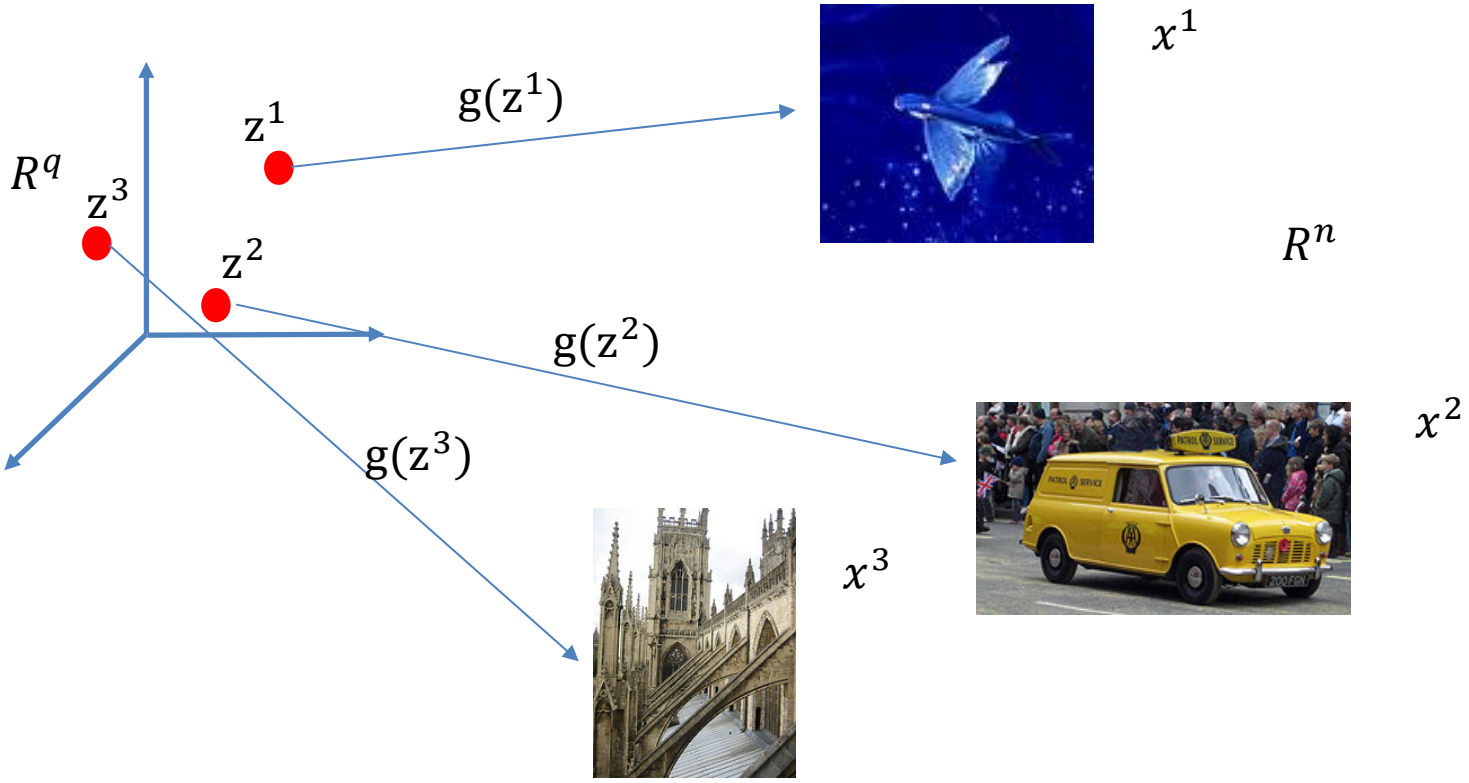
De Bezenac et al. 2021
Generating female images from male ones

Generative models

- Objective
  - General setup of deep generative models
    - Learn a generator nework $g_\theta: R^q \rightarrow R^n$ that transforms a latent distribution $\mathcal{Z} \subset R^q$ to match a target distribution $\mathcal{X}$
      - $\mathcal{Z}$ is usually a simple distribution e.g. Gaussian from which it is easy to sample, $q < n$
      - This is unlike traditional statistics where an analytic expression for the distribution is sought
    - Once trained the generator can be used for:
      - Sampling from the latent space:
        - $z \in R^q \sim \mathcal{Z}$ and then generate synthetic data via $g_\theta(.), g_\theta(z) \in R^n$
      - When possible, density estimation $p_\theta(x) = \int \; p_\theta(x|z)p_Z(z)dz$
        - with $p_\theta(x|z)$ a function of g_θ

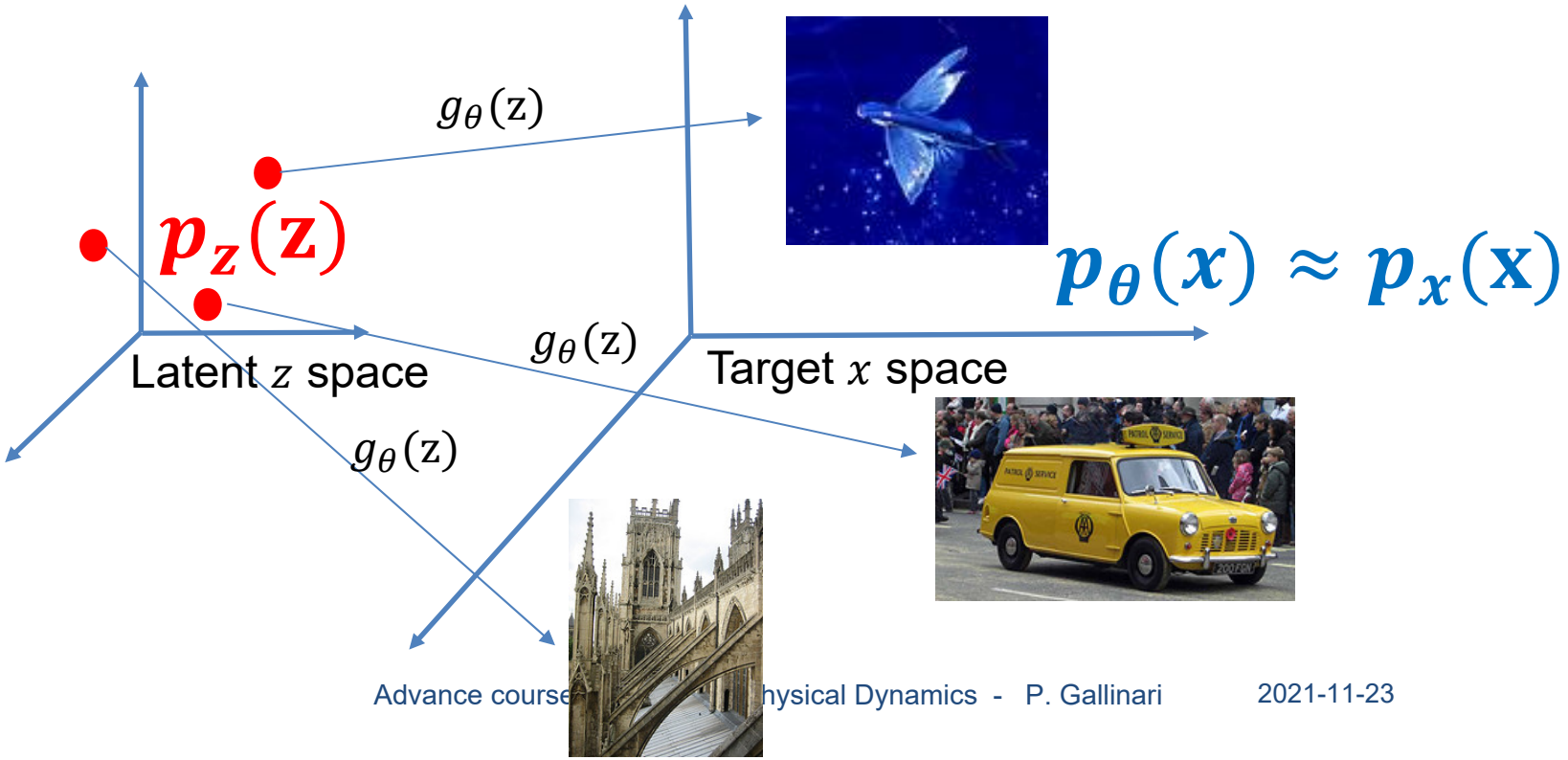
Generative models intuition

- Let $\{z^1, \dots, z^N\}, z^i \in R^q$ and $\{x^1, \dots, x^N\}, x^i \in R^n$, two sets of points in different spaces
  - Provided a sufficiently powerful model $g(x)$, it should be possible to learn complex deterministic mappings associating the two sets:



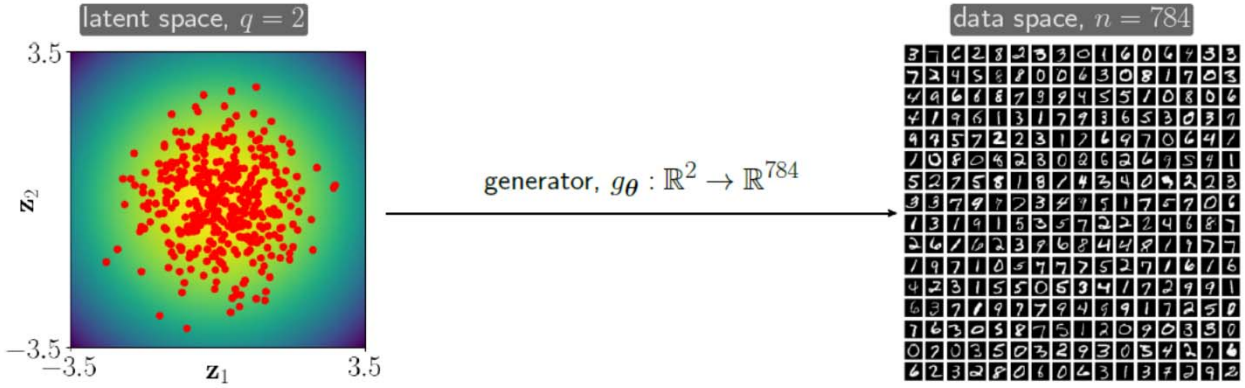
Generative models intuition

- Given distributions on a latent space $p_z(\mathrm{z})$, and on the data space $p_x(x)$, it is possible to map $p_z(\mathrm{z})$ onto $p_x(x)$
  - g_θ defines a distribution on the target space $p_x(g_\theta(z)) = p_\theta(x)$
    - $p_\theta(x)$ is the generated data distribution, objective: $p_\theta(x) \approx p_x(x)$
  - Data generation: sample $\mathrm{z} \sim Z$, transform with g_θ, $g_\theta(z)$

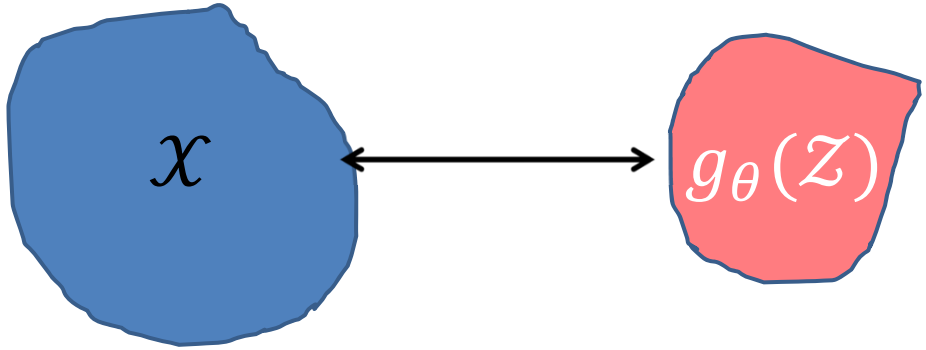


Generative models intuition

- Data generation: sample z$\sim Z$, transform with g_θ, $g_\theta(z)$



- Important issue
  - How to compare predicted distribution $p_\theta(x)$ and target distribution $p_X(x)$?



Course objective

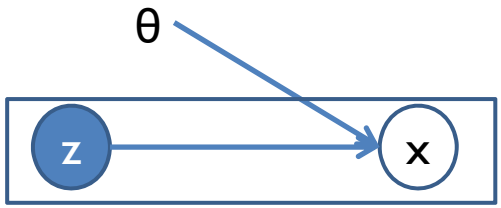
- Introduce three popular families of generative models
  - Joint requirements
    - Learn a generator g_θ from samples so that distribution $g_\theta(\mathcal{Z})$ is close to data distribution $\mathcal{X}, p_\theta(x) \approx p_x(x)$
    - This requires measuring the similarity between $g_\theta(\mathcal{Z})$ and $\mathcal{X}$
      - Different similarities are used for each family
- Three families
  - Generative Adversarial Networks
    - $g_\theta: R^q \to R^n, q \ll n$
    - Can approximate any distribution (no density hypothesis)
    - Similarity between generated and target distribution is measured via a discriminator or transport cost in the data space
  - Variational autoencoders
    - $g_\theta: R^q \to R^n, q \ll n$
    - Trained to maximize a lower bound of the samples' likelihood
    - Hyp: a density function explains the data
  - Flow models
    - $g_\theta: R^n \to R^n$ is diffeomorphic $q = n$, invertibilty simplifies the problem
    - Trained to maximize the likelihood of the samples
    - Hyp: a density function explains the data

Generative Adversarial Networks - GANs

Ian J. Goodfellow, et al. 2014

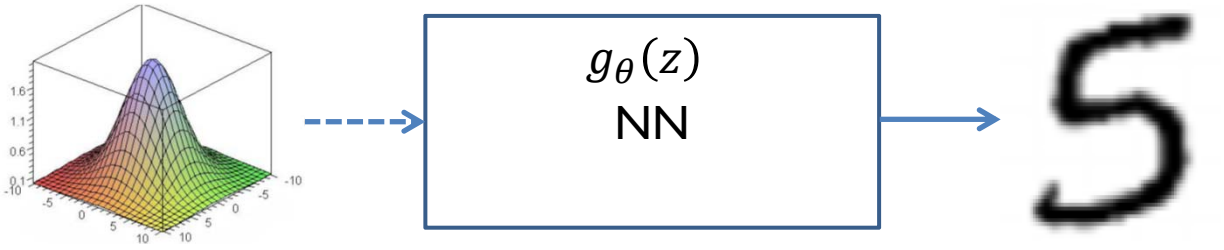
GANs

- Generative latent variable model



  - Given Samples $x^1, \ldots, x^N \in R^n$, with $x \sim \mathcal{X}$, latent space distribution $z \sim \mathcal{Z}$ e.g $z \sim \mathcal{N}(0, \mathrm{I})$, use a NN to learn a possibly complex mapping $g_\theta : R^q \rightarrow R^n$ such that:

$$p_x(g_\theta(z)) = p_\theta(x) \approx p_x(x)$$



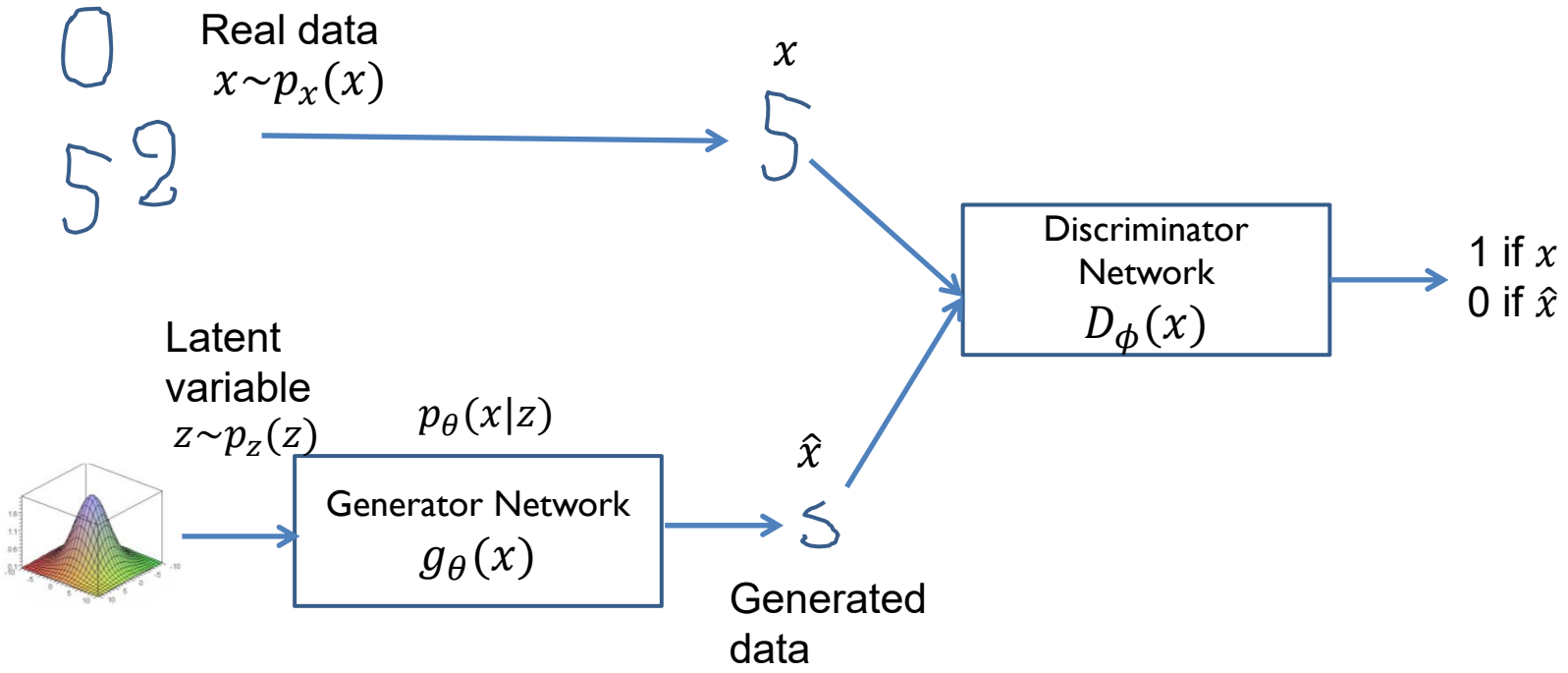
- Different solutions for measuring the similarity between $p_\theta(x)$ and $p_x(x)$
  - In this course: binary classification
- Note:
  - Once trained, sample from z directly generates the samples $g_\theta(z)$
  - Different from VAEs and Flows where the NN $g_\theta(.)$ generate distribution parameters

GANs – Adversarial training as binary classification

- Principle
  - A **generative** network generates data after sampling from a latent distribution
  - A **discriminant** network tells if the data comes from the generative network or from real samples
    - The discriminator will be used to measure the distance between the distributions $p_\theta(x)$ and $p_x(x)$
  - The two networks are trained together
    - The generative network tries to fool the discriminator, while the discriminator tries to distinguish between true and artificially generated data
    - The problem is formulated as a MinMax game
    - The Discriminator will force the Generator to be « clever » and learn the data distribution
- Note
  - No hypothesis on the existence of a density function
    - i.e. no density estimate (Flows), no lower bound (VAEs)

GANs – Adversarial training as binary classification
Intuition - Training

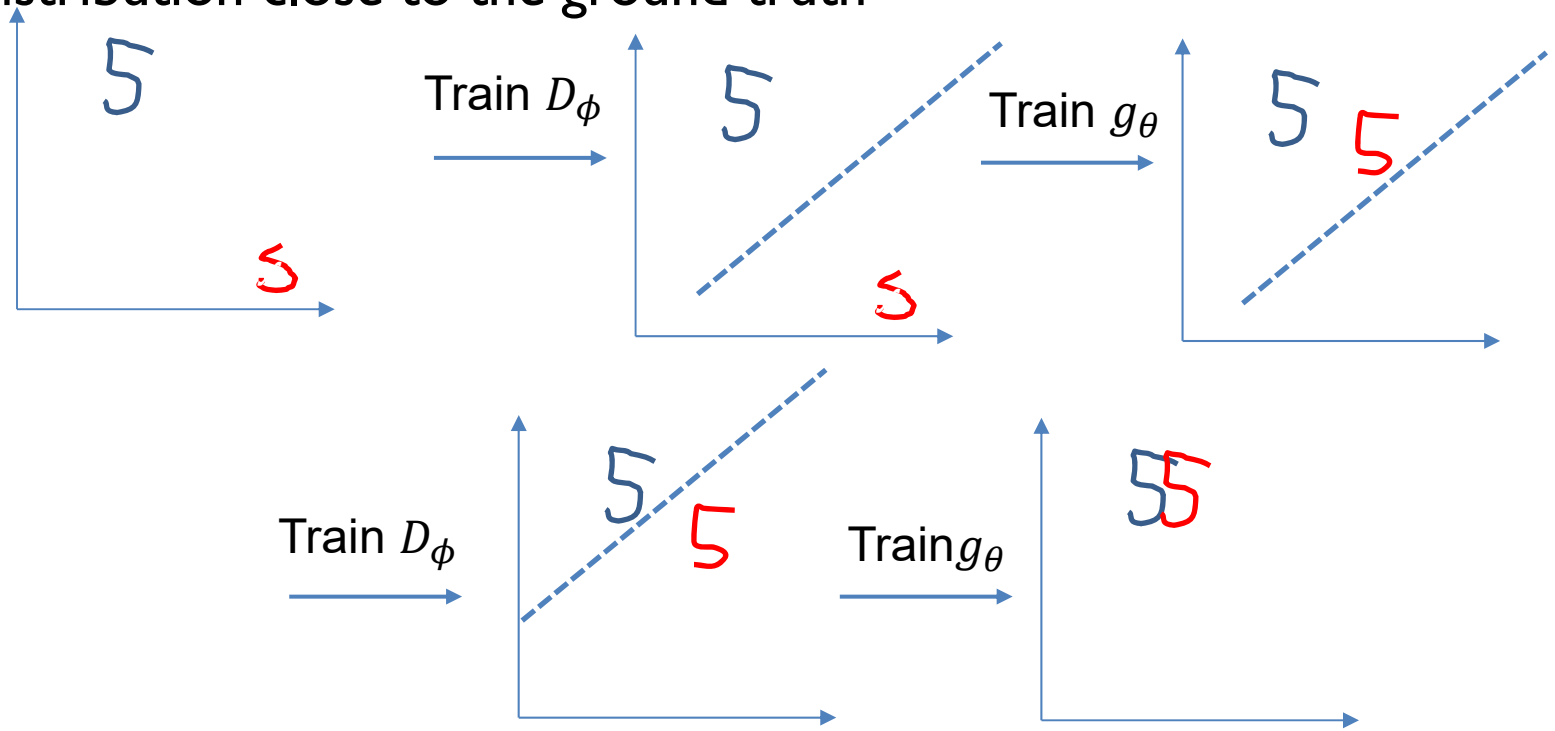
- Discriminator is presented alternatively with true (x) and fake ($\hat{x} = g_\theta(z)$)data



D_ϕ and g_θ are typically MLPs/Deep CNNs/…

GAN – Adversarial training as binary classification
Intuition - Training

- Algorithm alternates between optimizing D_ϕ (separate true and generated data) and g_θ (generate data as close as possible to true examples) – Once trained, G should be able to generate data witha distribution close to the ground truth



GANs - Adversarial training as binary classification
Loss function (Goodfellow et al. 2014)

- $x \sim p_x(x)$ distribution over data x
- $z \sim p_z(z)$ prior on z, usually a simple distribution (e.g. Normal distribution)
- Loss
  - $\min_{\theta} \max_{\phi} L(D_\phi, g_\theta) = E_{x \sim p_x(x)}[log D_\phi(x)] + E_{z \sim p_z(z)}[\log\left(1 - D_\phi(g_\theta(z))\right)]$
    - $g_\theta : R^q \rightarrow R^n$ mapping from the latent (z) space to the data (x) space
    - $D_\phi : R^n \rightarrow [0,1]$ probability that x comes from the data rather than from the generator g_θ
    - If g_θ is fixed, $L(D_\phi, g_\theta)$ is a classical binary cross entropy for D_ϕ, distinguishing real and fake examples
  - Note:
    - Training is equivalent to find D_{ϕ^*}, g_{θ^*} such that
      - $D_{\phi^*} \in arg \max_{\phi} L(D_\phi, g_{\theta^*})$ and $g_{\theta^*} \in arg \min_{\theta} L(D_{\phi^*}, g_\theta)$
      - Saddle point problem
        - instability
- Practical training algorithm
  - Alternates optimizing (maximizing) w.r.t. D_ϕ optimizing (minimizing) w.r.t. g_θ

GAN- Adversarial training as binary classification Equilibrium analysis (Goodfellow et al. 2014)

- The seminal GAN paper provides an analysis of the solution that could be obtained at equilibrium
- Let us define
  - $L(D_\phi, g_\theta) = E_{x\sim p_x(x)}[logD_\phi(x)] + E_{\mathrm{x}\sim p_\theta(x)}[\log\left(1 - D_\phi(x)\right)]$
    - with $p_x(x)$ the true data distribution and $p_\theta(x)$ the distribution of generated data
      - Note that this is equivalent to the $L(D, G)$ definition on the slide before
- If g_θ and D_ϕ have sufficient capacity
  - Computing $\underset{\theta}{argmin}$
    - $g^* = \underset{\theta}{argmin} \max_\phi L\left(D_\phi, g_\theta\right)$
  - Is equivalent to compute
    - $g^* = argmin_\theta D_{JS}(p_x, p_\theta)$ with $D_{JS}(,)$ the Jenson-Shannon dissimilarity measure between distributions
    - The loss function of a GAN quantifies the similarity between the real sample distribution and the generative data distribution by JSD when the discriminator is optimal
  - If the optimum is reached
    - $D(x) = \frac{1}{2}$ for all $x \rightarrow$ Equilibrium

Adversarial training as binary classification
Training GANs

- Training alternates optimization (SGD) on D_ϕ and g_θ
  - In the alternating scheme, g_θ usually requires more steps than D_ϕ+ different batch sizes
- It is known to be highly unstable with two pathological problems
  - Oscillation: no convergence
  - Mode collapse: G collapses on a few modes only of the target distribution (produces the same few patterns for all z samplings)
  - Low dimensional supports (Arjovsky 2017): $p_x$and p_θ may lie on low dimensional manifold that do not intersect.
    - It is then easy to find a discriminator, without p_θ close to p_x
  - Lots of heuristics, lots of theory, but
    - Behavior is still largely unexplained, best practice is based on heuristics

GANs examples
Deep Convolutional GANs (Radford 2015 - historical example) - Image generation

- LSUN bedrooms dataset - over 3 million training examples –

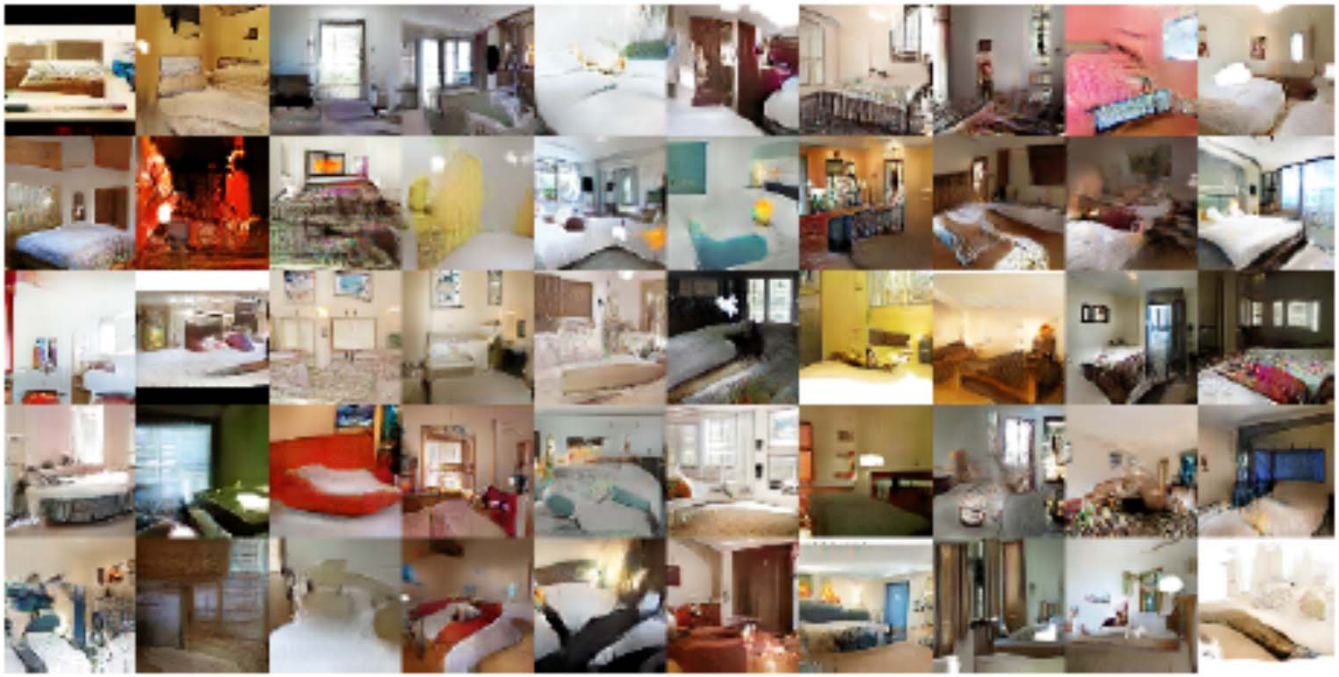
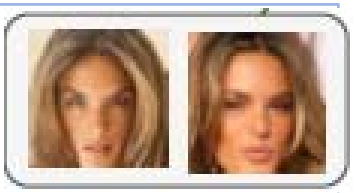


Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

Fig. Radford 2015

Gan example
MULTI-VIEW DATA GENERATION WITHOUT VIEW SUPERVISION (Chen 2018)



- Objective
  - Generate images by disantangling content and view
    - e.g. content 1 person, View: position, illumination, etc
  - 2 latent spaces: view and content
    - Generate image pairs: same item with 2 different views
    - Learn to discriminate between generated and real pairs

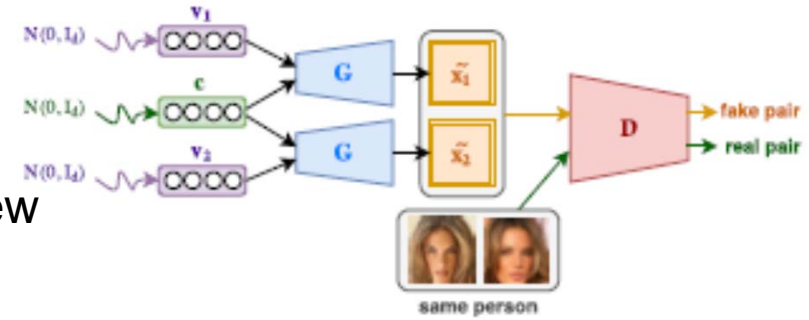
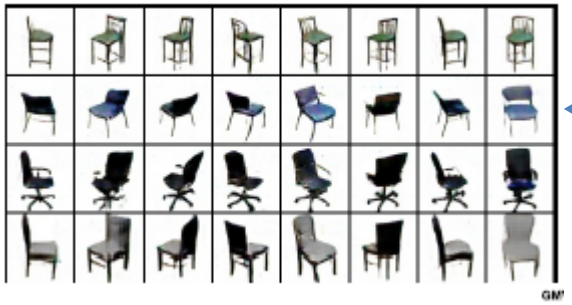


Fig. Chen 2018

Column = view

Column = view



1 row = 1 content

Conditional GANs (Mirza 2014)

- The initial GAN models distributions by sampling from the latent Z space
- Many applications require to condition the generation on some data
  - e.g.: text generation from images, in-painting, super-resolution, etc
- Conditional GANs
  - Both the generator and the discriminator are conditioned on variable y – corresponding to the conditioning data

$$\min_{\theta}\max_{\phi} V(D,g) = E_{x\sim p_x(x)}[logD_\phi(x|y)] + E_{z\sim p(z)}[\log\Big(1 - D_\phi(g_\theta(z|y))\Big)]$$

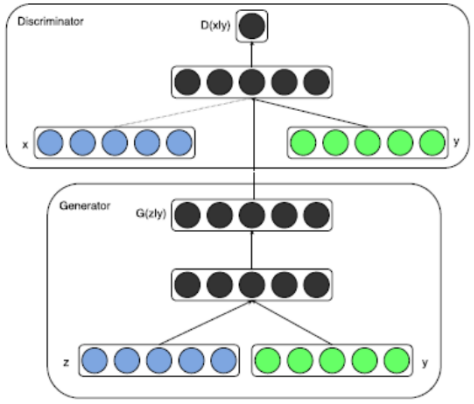


Fig. (Mirza 2014)

Conditional GANs example
Generating images from text (Reed 2016)

- Objective
  - Generate images from text caption
  - Model: GAN conditioned on text input
- Compare different GAN variants on image generation
- Image size 64x64

Fig. from Reed 2016

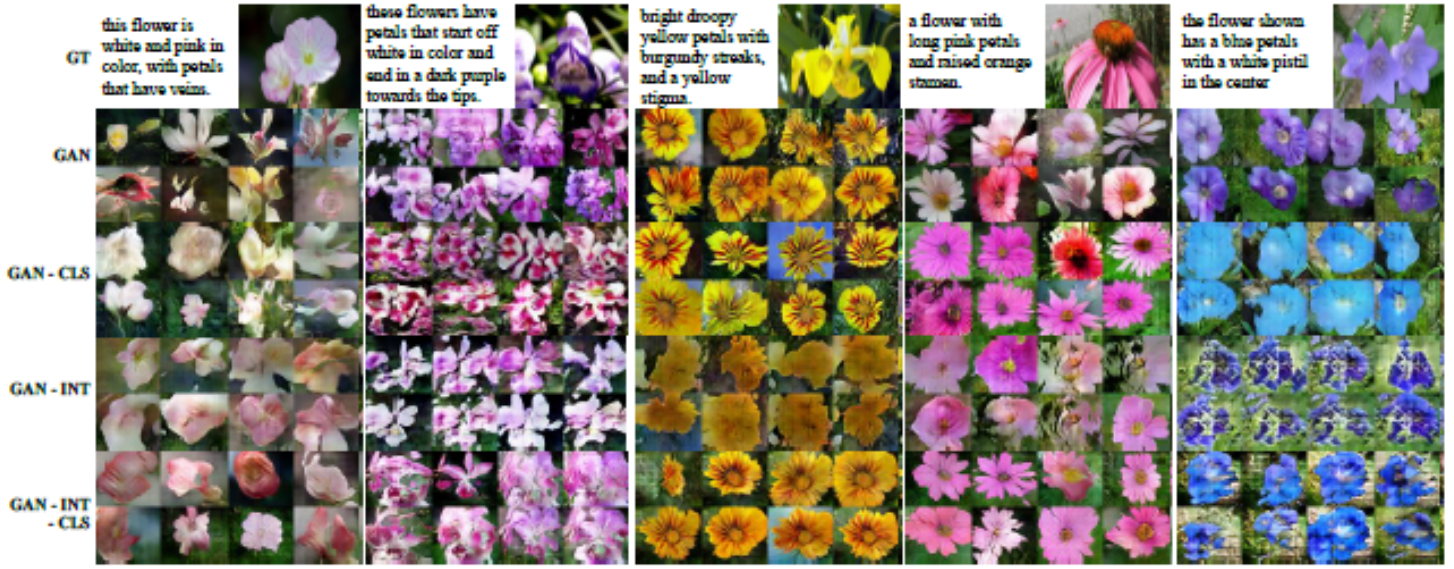


Figure 4. Zero-shot generated flower images using GAN, GAN-CLS, GAN-INT and GAN-INT-CLS. All variants generated plausible images. Although some shapes of test categories were not seen during training (e.g. columns 3 and 4), the color information is preserved.

Cycle GANs (Zhu 2017)

- Objective
  - Learn to « translate » images without aligned corpora
    - 2 corpora available with input and output samples, but no pair alignment between images
  - Given two unaligned corpora, a conditional GAN can learn a correspondance between the two distributions (by sampling the two distributions), however this does not guaranty a correspondance between input and output
- Approach
  - (Zhu 2017) proposed to add a « consistency » constraint similar to back translation in language
    - This idea has been already used for vision tasks in different contexts
    - Learn two mappings
      - $G: X \rightarrow Y$ and F: $Y \rightarrow X$ such that:
      - $F \circ G(x) \simeq x$ and G $\circ F(y) \simeq$y
      - and two discriminant functions $D_Y$and D_X

Cycle GANs (Zhu 2017)

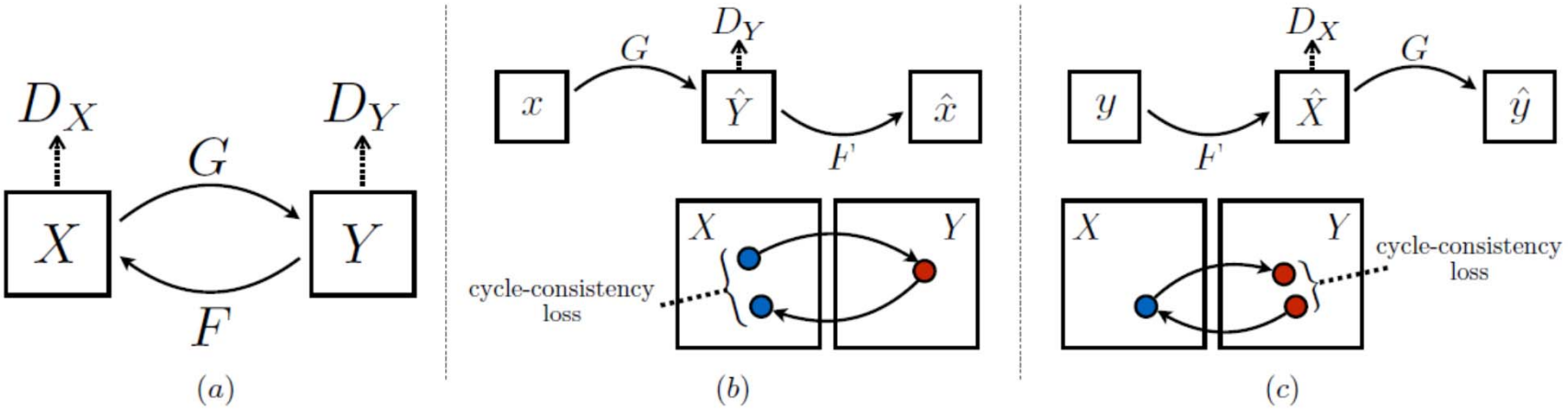
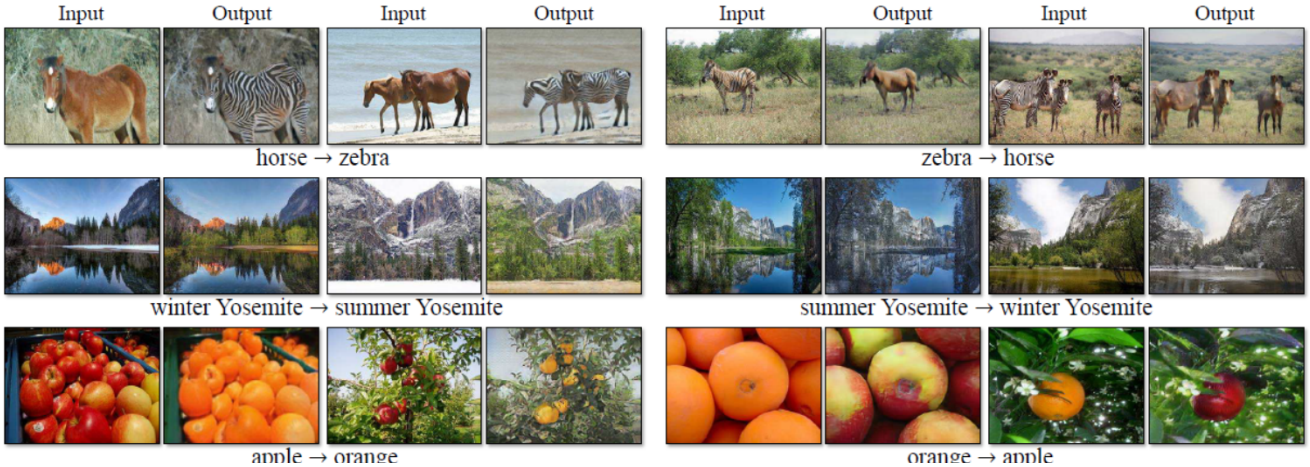


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X, F, and X. To further regularize the mappings, we introduce two "cycle consistency losses" that capture the intuition that if we translate from one domain to the other and back again we should arrive where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Fig (Zhu 2017)

Cycle GANs (Zhu 2017)

- Training
  - The loss combines two conditional GAN losses (G, D_Y) and (F, D_X) and a cycle consistency loss
  - $C_{cycle}(F, G) = E_{p_{data}(x)}[\|F(G(x)) - x\|_1] + E_{p_{data}(y)}[\|G(F(y)) - y\|_1]$
  - $C = L(G, D_Y) + L(F, D_X) + C_{cycle}(F, G)$
  - Note: they replaced the usual $V(G, D_Y)$ and $V(F, D_X)$ term by a mean square error term, e.g.:
    - $\mathrm{L}(G, D_Y) = E_{pdata(y)}[(D_Y(y) - 1)^2] + E_{data(x)}[D_Y(G(x))]$



Geophysical example (1)
Nowcasting (Ravuri et al. 2021) – DeepMind & MET office UK

- Precipitation nowcasting
  - Predict precipitations (rainfall) up to 2 hours ahead
- Classical methods
  - Precipitation is modeled following the advection equation with a radar source term
    - Motion field is estimated using optical flow
    - Uncertainty modeling via stochastic perturbations added to the motion field and intensity model
    - This provides both stochastic and deterministic forecasts
- Paper
  - Compares GAN based prediction with alternatives

Geophysical example (1)
Nowcasting (Ravuri et al. 2021) – DeepMind & MET office UK - Model

- Objective
  - Given M contextual radar field estimates of surface precipitation, predict the next N radar fields
    - $P(X_{M+1:M+N}|X_{1:M}) = \int P(X_{M+1:M+N}|Z, X_{1:M})P(Z|X_{1:M})\, dZ$
      - With Z a latent variable – in practice Monte Carlo estimates are used for approximating the integral
- Model
  - Conditional GAN
    - Context $M = 4$ frames(20 mn), Prediction $N = 18$ frames (90 mn)
  - Loss function - 3 terms
    - Spatial discriminator: Convolutional NN, distinguishes invidual generated frames from bserved frames, ensures spatial consistency
    - Temporal discriminator: 3D convolutional NN, distinguishes between observed and generated radar sequences, temporal consistency
    - Regularization term: Penalizes the deviations at the grid cell resolution level between mean prediction and real radar sequence
- Data
  - Sequences of 256x256 crops extracted from radar stremas of length 110 mn (22 frames)
  - Training years 2016-2018, testing 2019

Geophysical example (1)
Nowcasting (Ravuri et al. 2021) – DeepMind & MET office UK Model

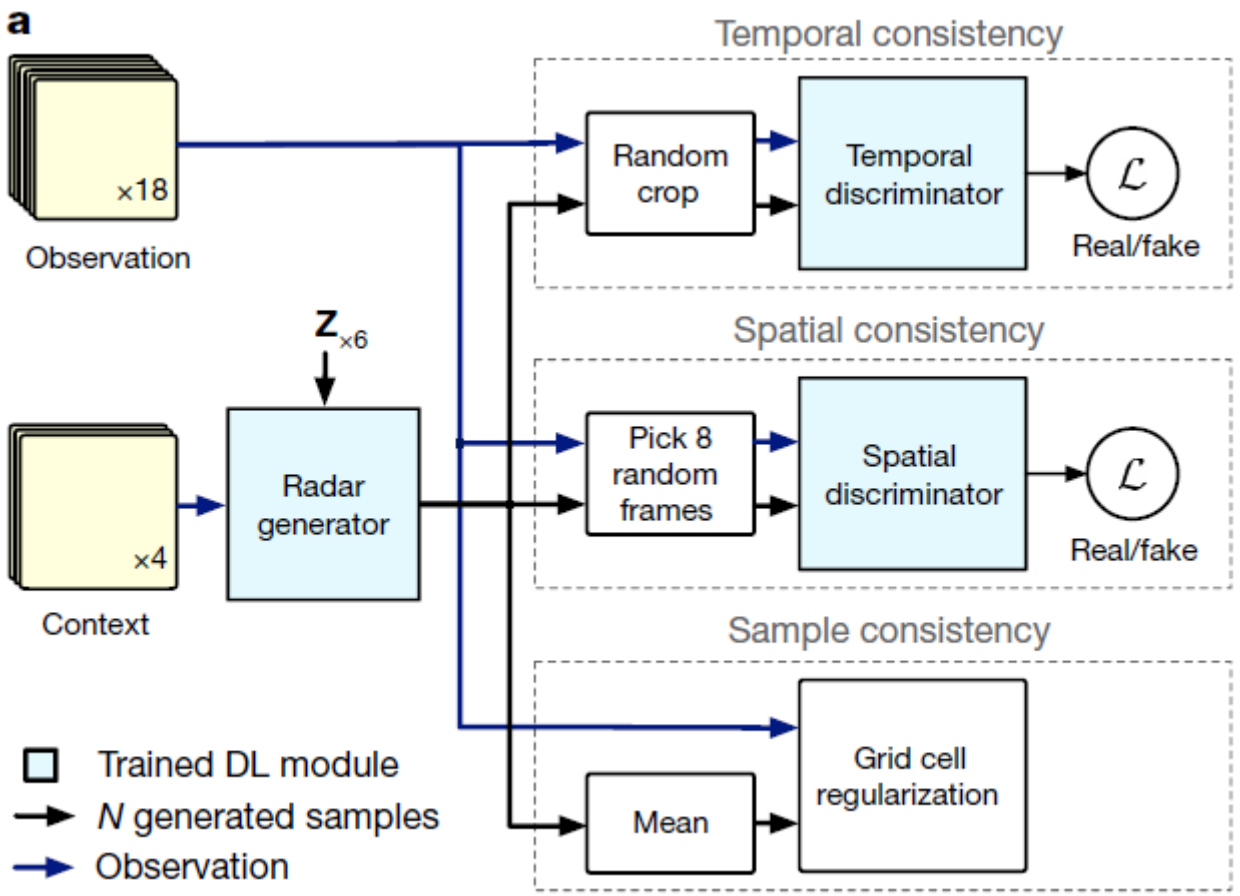


Fig. Ravuri et al. 2021

Geophysical example (1)
Nowcasting (Ravuri et al. 2021) – DeepMind & MET office UK-examples

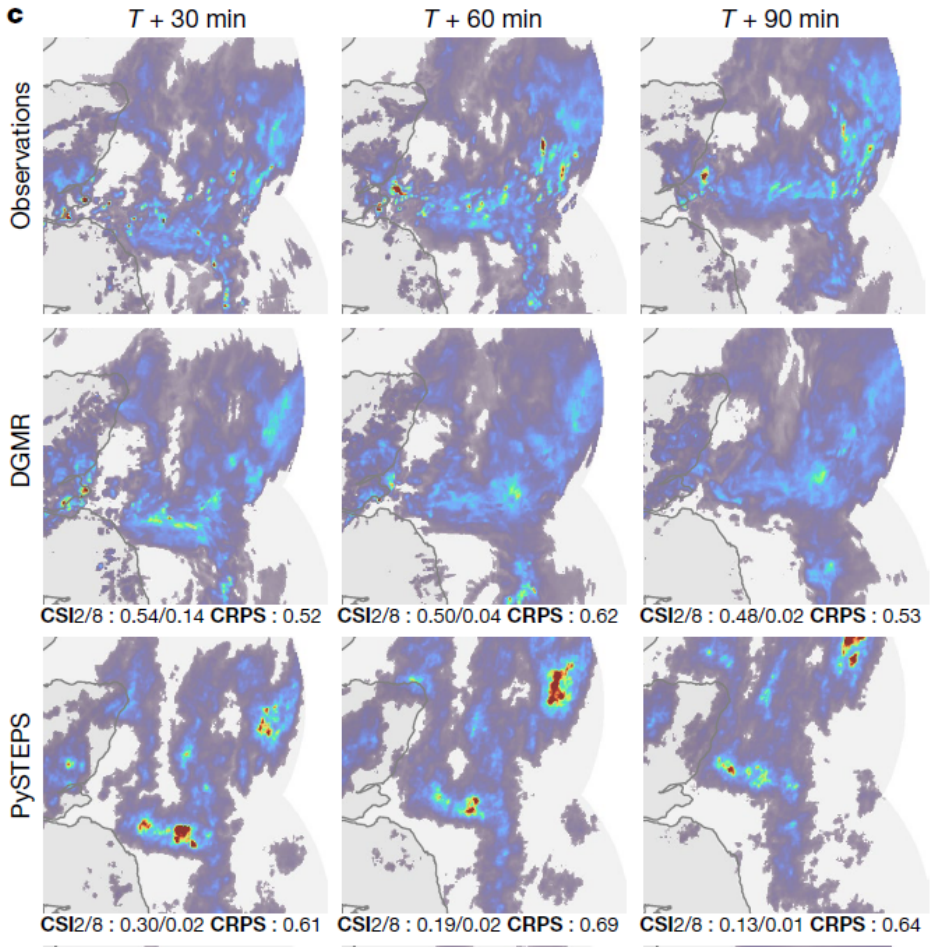


Fig. Ravuri et al. 2021

GAN model

SOTA physical model

Qualitative evaluation by human experts. GAN about 90% first choice over 3 alternative models

Geophysical example (2)
Downscaling (Leinonen et al. 2020)

- Downscaling
  - Predict high resolution maps (e.g. 1km) from low resolution (e.g. 100 km) maps usually coming from global climate models
    - Close to super-resolution problems in vision (image or videos)
      - Example SRGAN (Ledig et al. 2018) 4xupscaling:

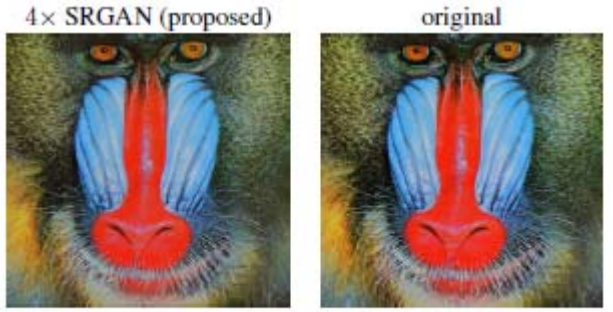


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Fig. Ledig et al. 2018

- Paper
  - Stochastic downscaling for time-evolving data: generate multiple high resolution sequences from low resolution ones
  - Applications
    - Radar measured precipitations
    - Cloud optical thickness

Geophysical example (2)
Downscaling (Leinonen et al. 2020)- model

- Conditional GAN
  - Relatively complex architecture includes convolutional and recurrent NNs
  - Generator
    - Input 8x8x8x1 tensor (8 time steps, image 8x8, 1 variable (e.g. rainfall)) corresponding to low resolution + noise concatenated to each 8x8 lr image
    - Output high resolution 8x128x128x1 tensor – i.e. corresponding high resolution sequence
  - Discriminator
    - Input: 8x8x8x1low resolution image concatenated with 8x128x128x1 high reolution
      - Objective
        - $Min_{\phi} E_{x,y,z}[L_D(x, y, z; \phi]$ with x h.r. sequences, y l.r. sequences, z noise
    - Output: 1/0 - does the pair belongs to the training set, i.e. do the two images correspond (low and high resolution) or is the hr one a generated fake?
      - Objective
        - $Min_{\theta} E_{y,z}[L_g(y, z; \theta)]$ with $L_g(y, z; \theta) = D_{\phi}(G_{\theta}(y, z))$

Geophysical example (2)
Downscaling (Leinonen et al. 2020)- model

Fig. Leinonen et al. 2020

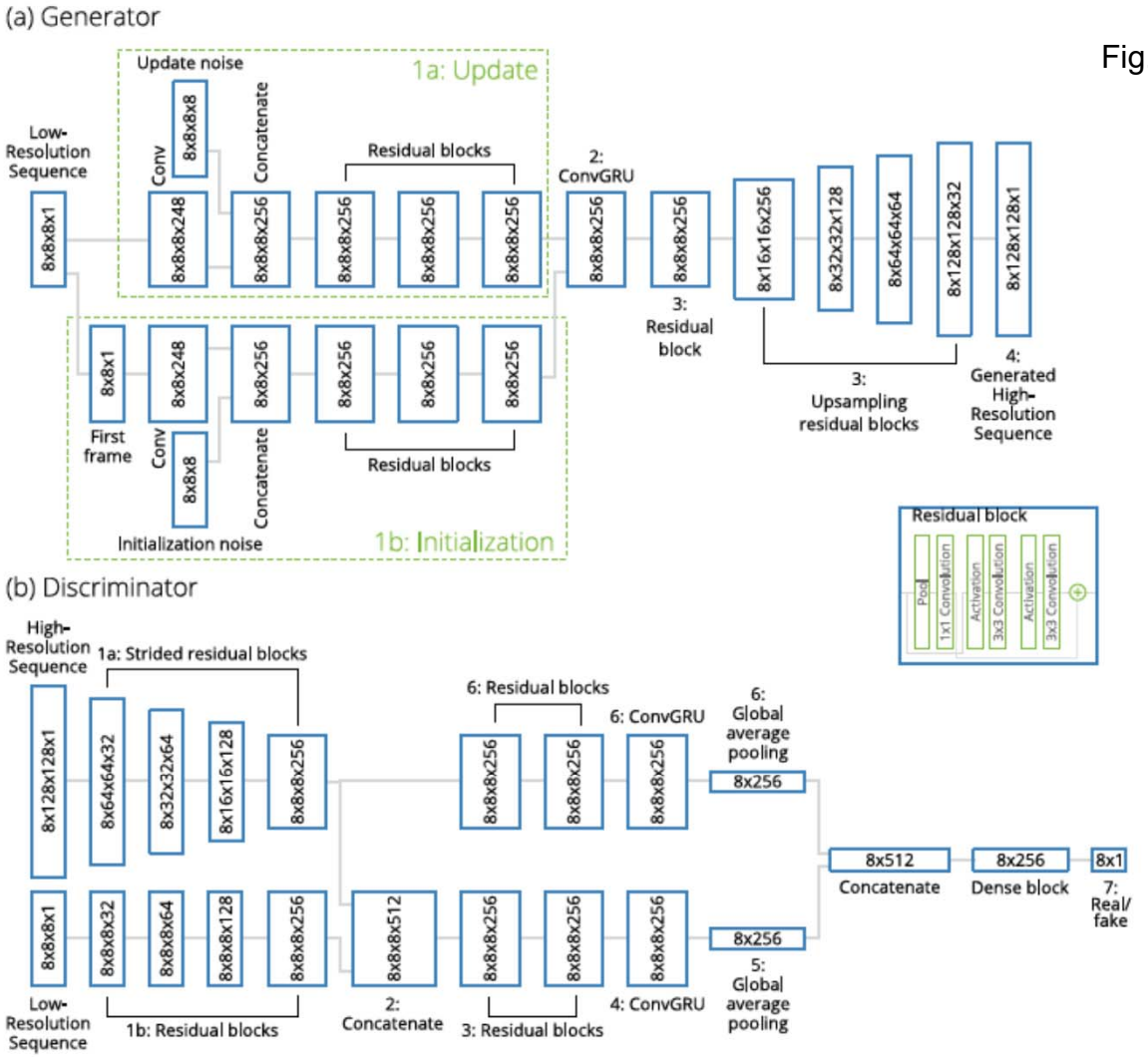


Fig. 1. The architectures of (a) the generator and (b) the discriminator. The numbered labels correspond to the descriptions in Section II-B. The dimensions shown here are for the training configuration where high-resolution image size $h \times w = 128 \times 128$, the number of frames per sequence $N_t = 8$ and the number of variables $N_v = 1$ for both datasets considered here. After training, the network can be evaluated using different values of these parameters.

Geophysical example (2)
Downscaling (Leinonen et al. 2020)- example – rain radar data

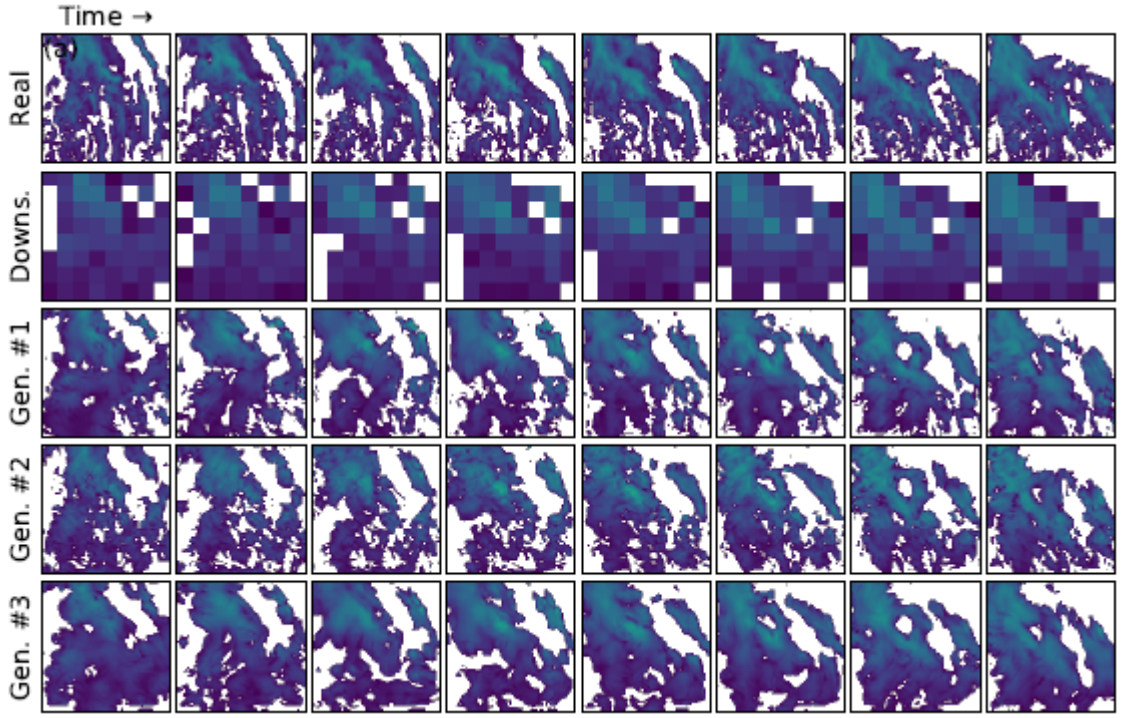
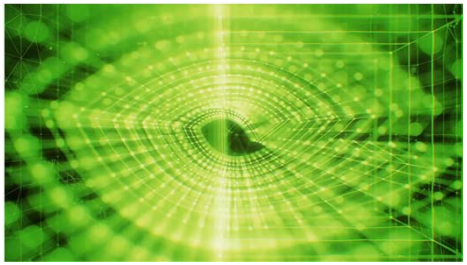


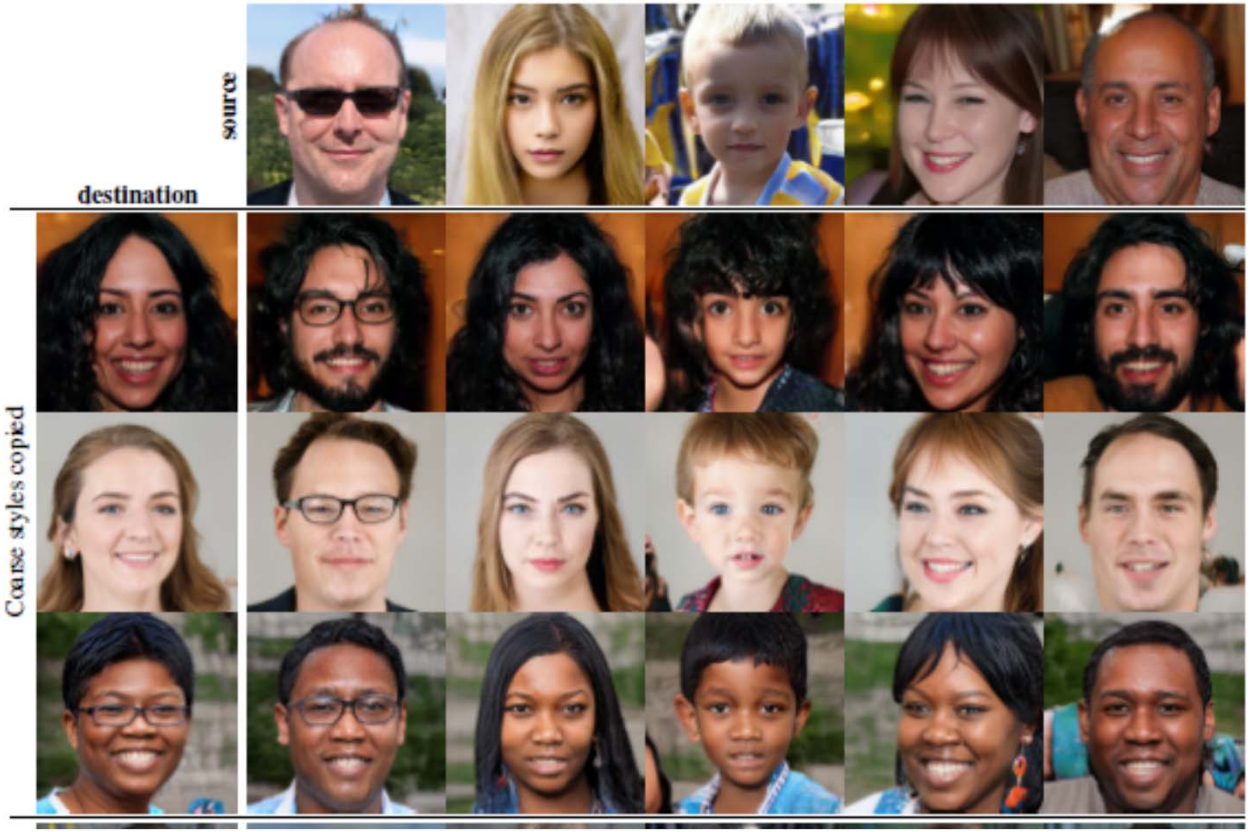
Fig. Leinonen et al. 2020

Fig. 2. Examples of reconstructed image sequences from the MCH-RZC test dataset. Each main panel (a)–(c) shows the real high-resolution image on the top row, the downsampled version on the second row, and three examples of reconstructions created by the GAN on the last three rows.

Large size demos
Style GAN (Karras et al. 2019)

- (Karras et al. 2019 - NVIDIA) – Style GAN





GANs summary

- Likelihood free training
- Optimize quality of generated samples
- Difficult to train
  - arXiv (2021-07-16) : **Showing 1–50 of 5,211 results for all: gans**
  - Mode collapse, convergence, ..
  - No best method, relies on heuristics (e.g. normalization)
  - Works on very high dimensional spaces

Variational Auto-Encoders

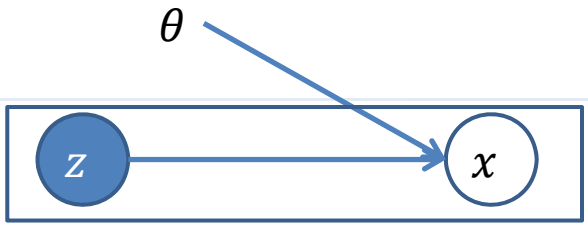
After Kingma D., Welling M., Auto-Encoding Variational Bayes, ICLR 2014

Prerequisite KL divergence

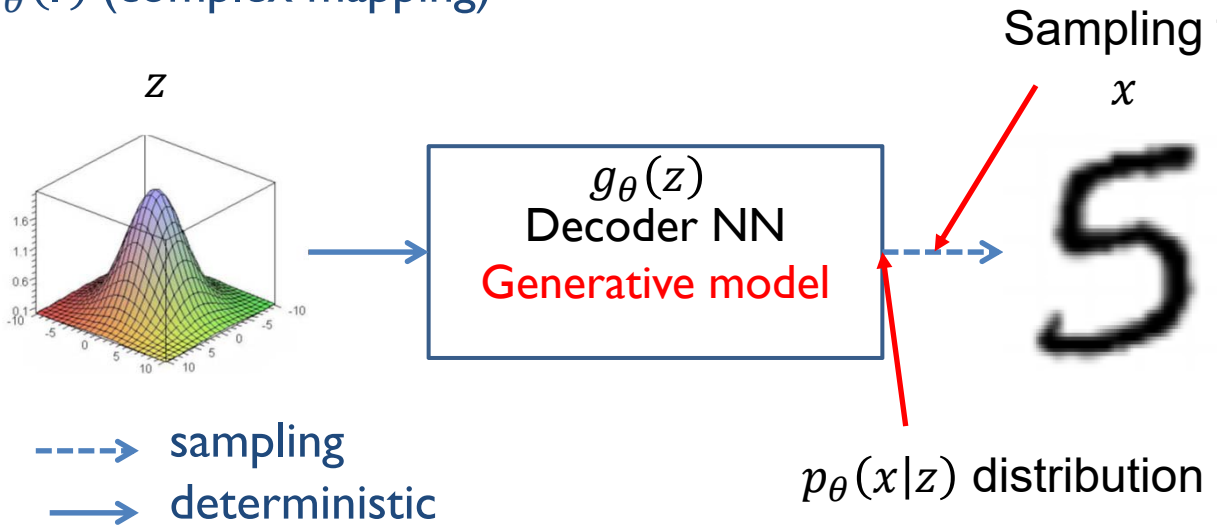
- Kullback Leibler divergence
  - Measures of the difference between two distributions p and q
  - Continuous variables
    - $D_{KL}(p(y)||q(y)) = \int_y (\log\frac{p(y)}{q(y)})p(y)dy$
  - Discrete variables
    - $D_{KL}(p(y)||q(y)) = \sum_i (\log\frac{p(y_i)}{q(y_i)})p(y_i)$
- Property
  - $D_{KL}(p(y)||q(y)) \geq 0$
  - $D_{KL}(p(y)||q(y)) = 0$ iff $p = q$

note: D_{KL} is asymmetric, symmetric versions exist, e.g. Jensen-Shannon divergence

VAEs – Intuition - Generator



- Generative latent variable model
- Sample from a simple distribution $z \sim p(z)$, and generate $p_\theta(x|z)$
- Training via Maximum Likelihood (approximate)
  - The parameters distribution will be learned with a NN generator $g_\theta(.)$ (complex mapping)



- $g_\theta(.)$ generate the distribution parameters, not the samples as for GANs

VAEs – Intuition - generator

- Intuitively, z might correspond to the hidden factors conditioning the generation of the data: (digit, angle) for MNIST, (pose, expression) for the faces

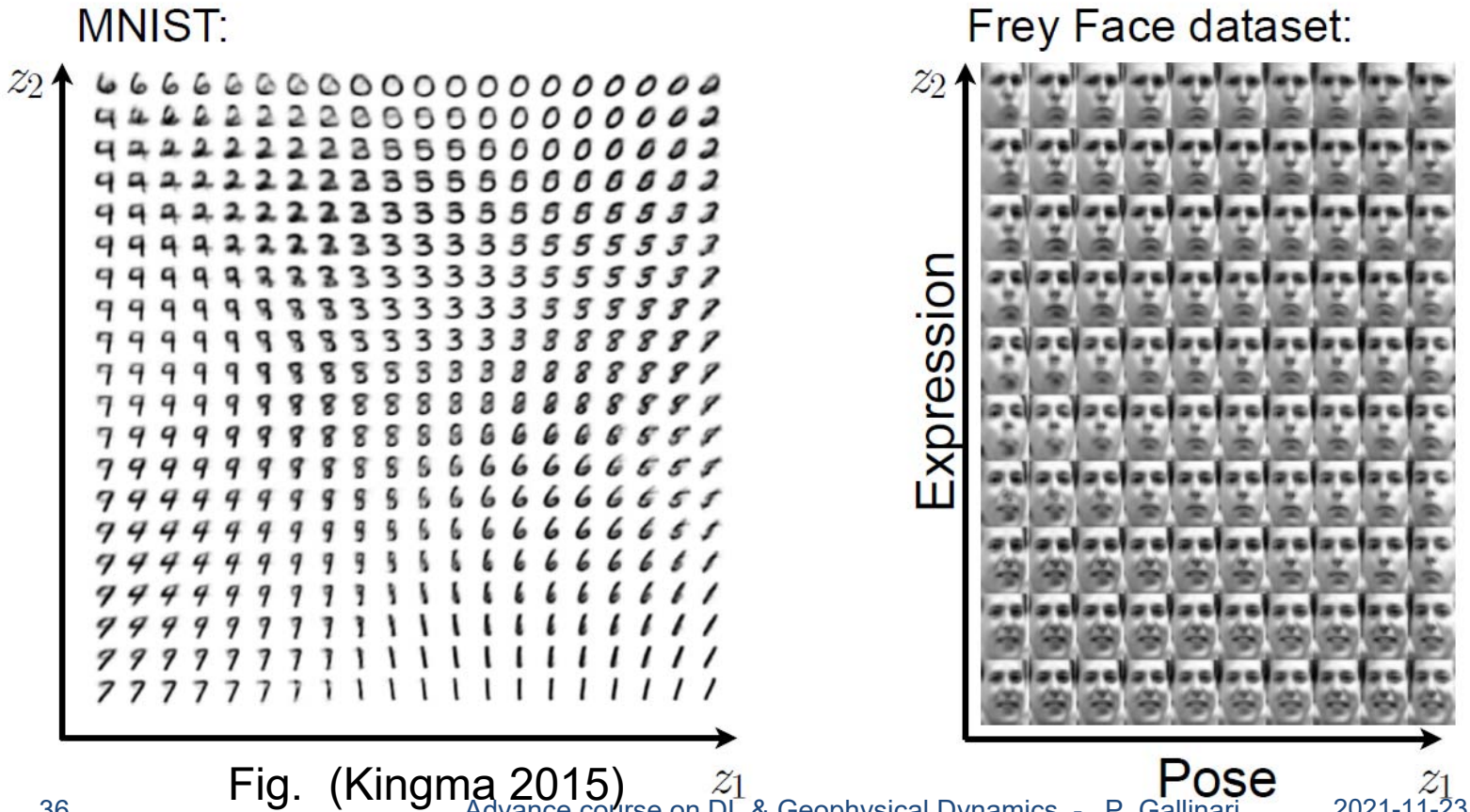
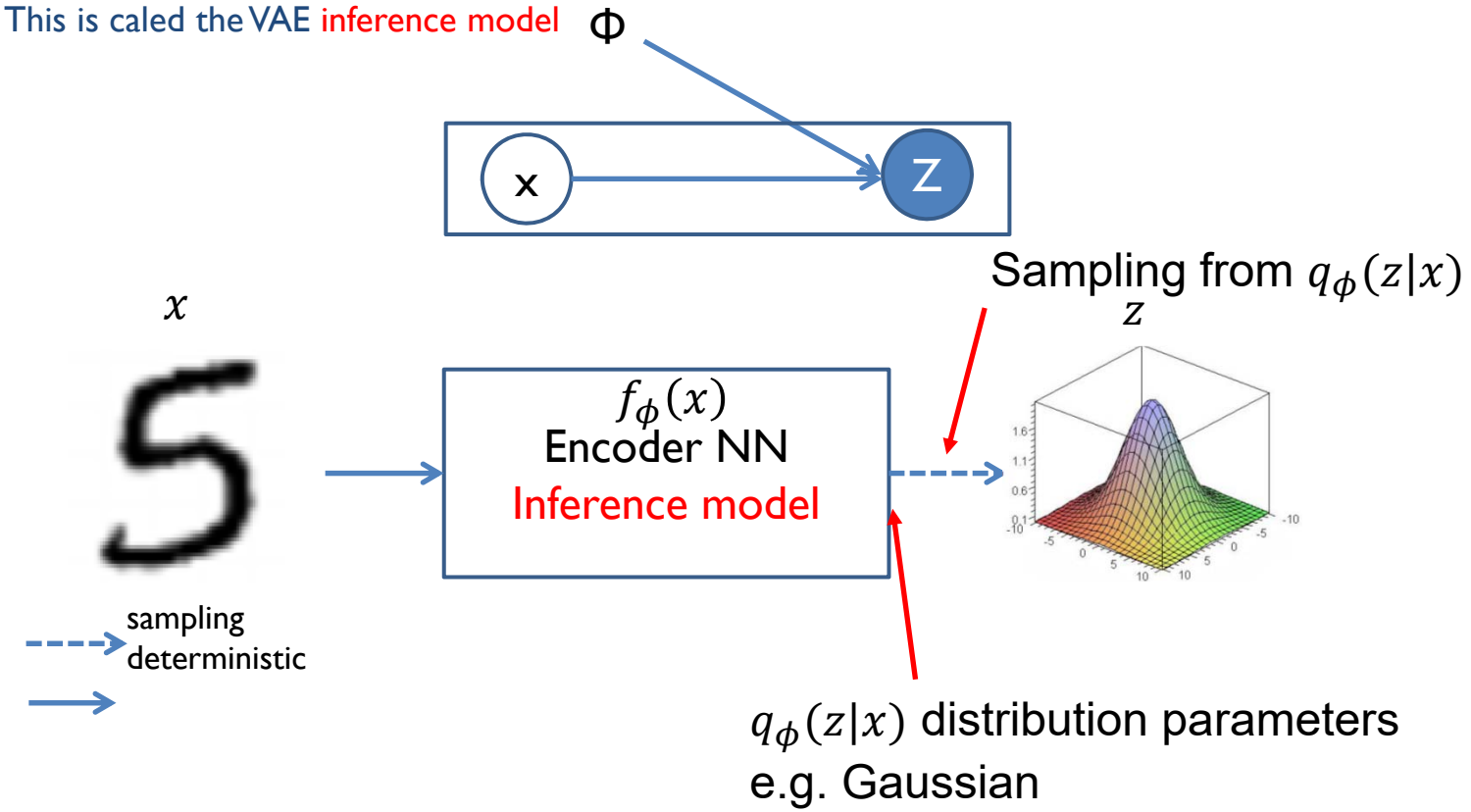


Fig. (Kingma 2015)

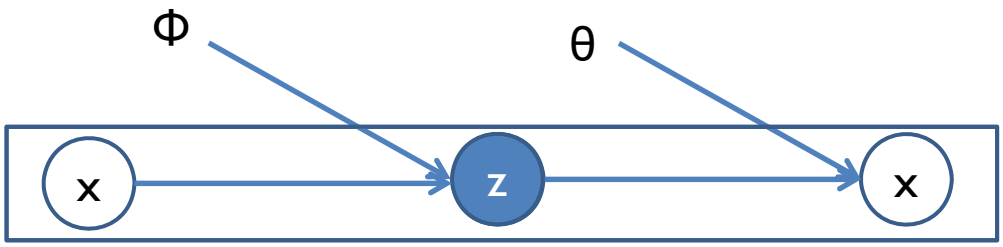
VAEs – Intuition – Training - Inference model

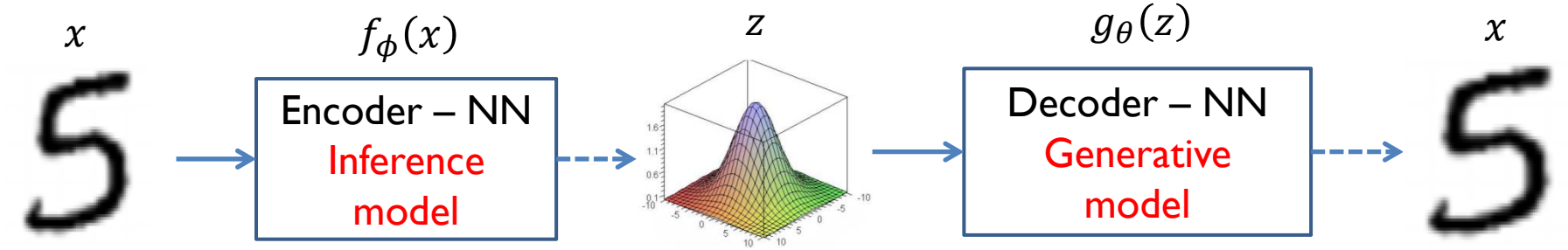
- Training
  - Maximum likelihood
  - When the x space is large, with $z \sim \mathcal{N}(0, \mathrm{I})$, most $p(x|z)$ will be nearly 0 - not relevant for ML training
  - How to choose an adequate distribution for z?
  - VAEs compute this distribution $q_\phi(z|x)$ for z using a parametric function $\mathrm{f}_\phi(x)$
  - This is caled the VAE inference model



VAEs – Intuition - Training

- Putting it all together: VAE model for training





- ⇢ sampling
- → deterministic

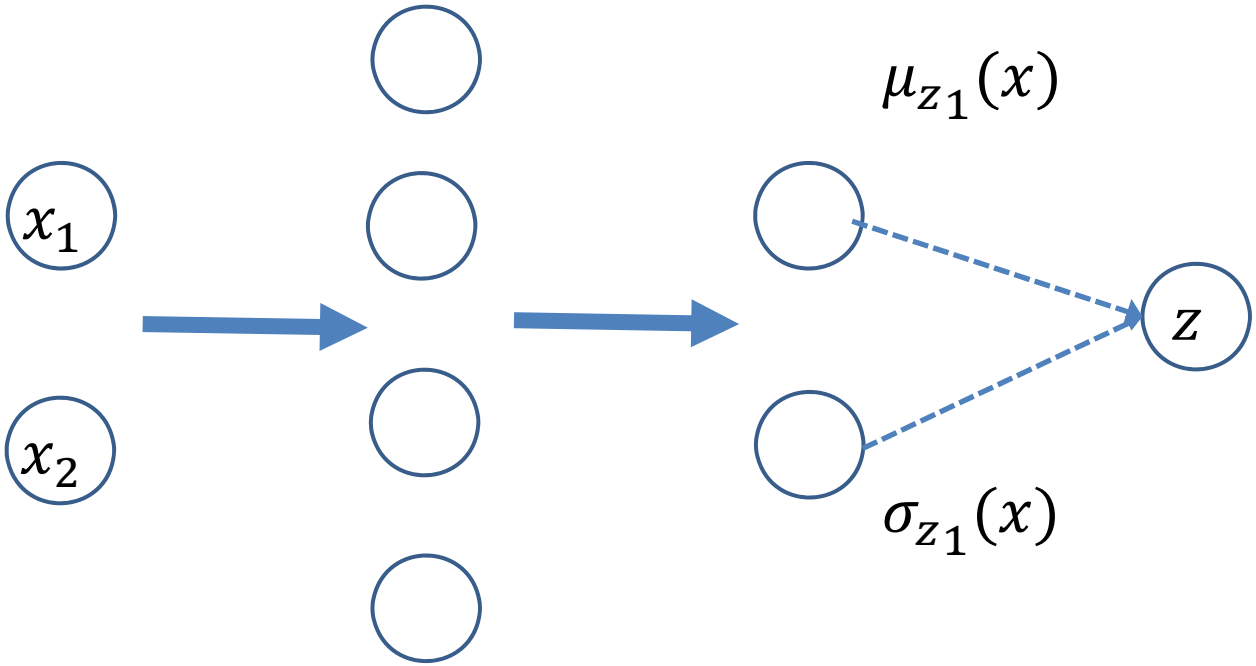
Note: both the encoder and the decoder are stochastics

VAE- illustration
Exemple: Gaussian priors and posteriors

- Special case: gaussian priors and posteriors
- Hyp:
  - $p(z) = \mathcal{N}(0, I)$
  - $p_\theta(x|z) = \mathcal{N}(\mu(z), \sigma(z)), \sigma(z)$ diagonal matrix, $x \in R^D$
  - $q_\phi(z|x) = \mathcal{N}(\mu(x),\ \sigma(x)), \sigma(x)$ diagonal matrix, $z \in R^J$

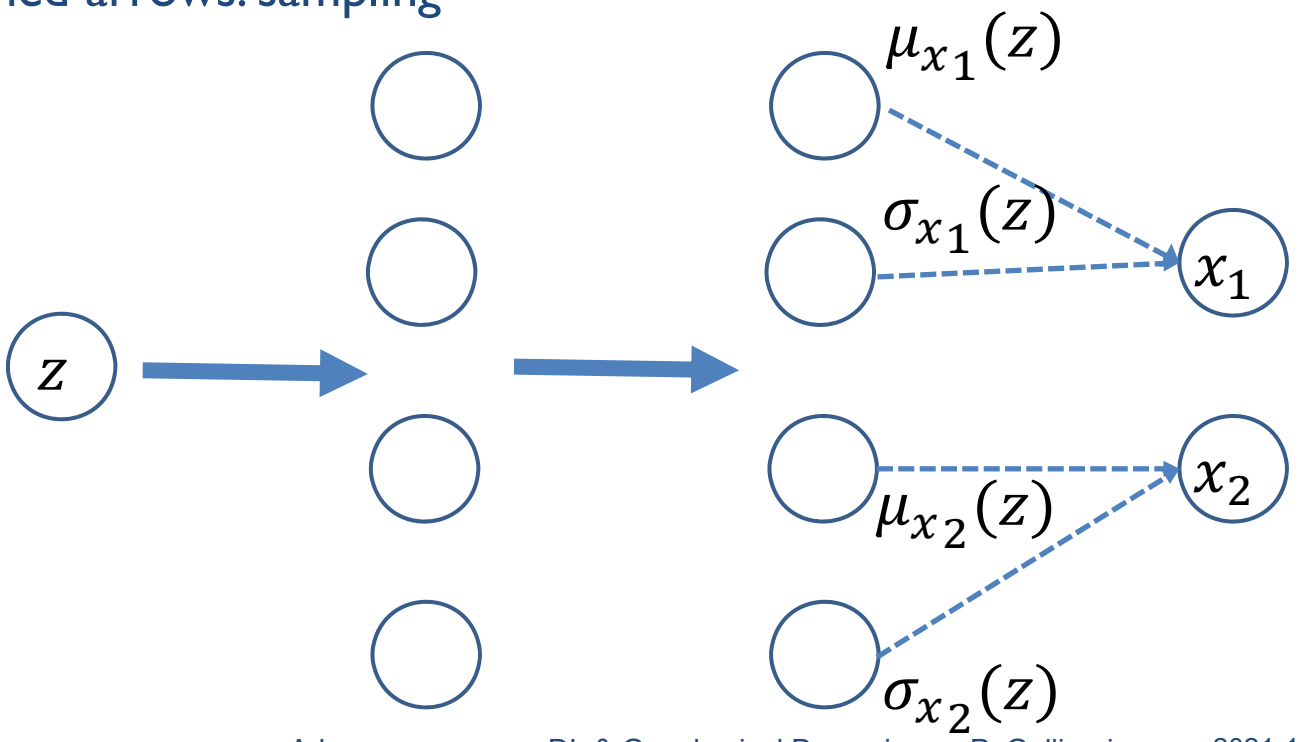
VAE- illustration
Gaussian priors and posteriors

- Encoder
  - in the example z is 1 dimensional and x is 2 dimensional, f is a 1 hidden layer MLP with linear output units and tanh hidden units
  - full arrows: deterministic
  - dashed arrows: sampling



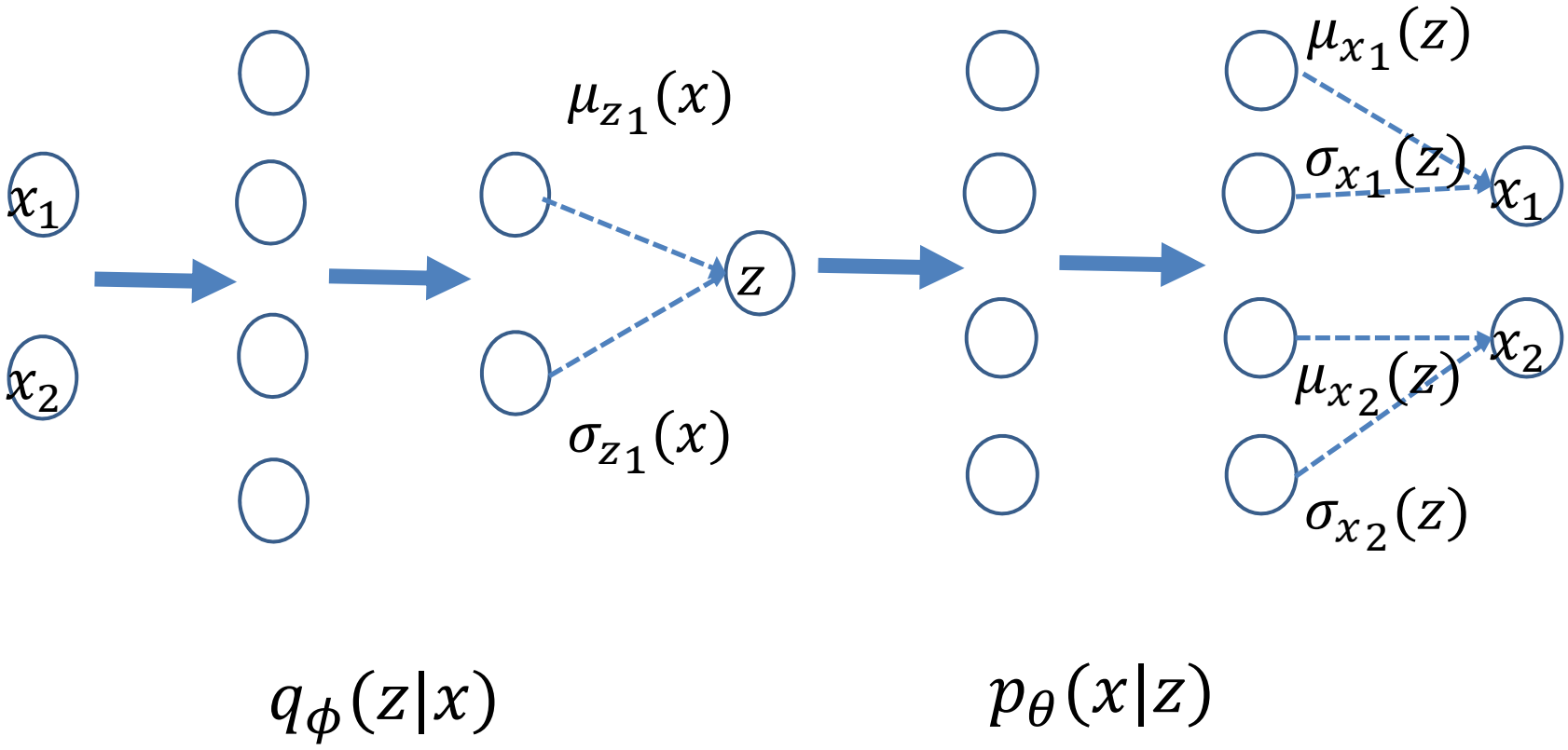
VAE- illustration
Exemple: Gaussian priors and posteriors

- Decoder:
  - in the example z is 1 dimensional and x is 2 dimensional, g is a 1 hidden layer MLP with linear output units and tanh hidden units
  - full arrows: deterministic
  - dashed arrows: sampling



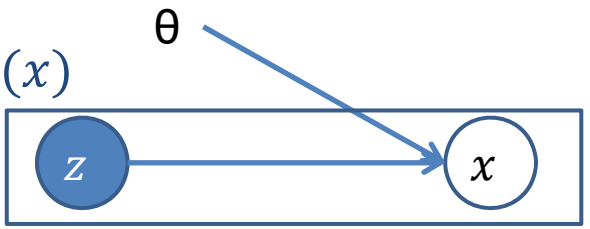
VAE
Gaussian priors and posteriors - Illustration

- Putting it all together



VAEs - Description

- Let us suppose available
  - A set of observed variables x with density $p_x(x)$



- Training
  - We want to learn a latent model, i.e. a joint pdf $p_\theta(x, z)$
    - θ model parameters, p_θ continuous and differentiable
  - Objective: optimize θ so that $g_\theta(z)$ produces samples in the x space with a high probability,
    - i.e. maximize the evidence (likelihood of the model)
    - $p_\theta(x) = \int \; p_\theta(x, z)dz = \int \; p_\theta(x|z)p_\theta(z)dz$
  - Typycally, both the prior $p_\theta(z)$ and the posterior $p_\theta(x|z)$ may be simple distributions, here we will consider gaussians
    - If $g_\theta(z)$ is a NN, the marginal $p_x(x)$ may be arbitrarily complex even with simple priors $p_\theta(z)$ and posteriors $p_\theta(x|z)$

VAEs – Description - Training

- In order to maximize the evidence, one has to deal with two problems
  - Define the latent variable distribution $p_\theta(z)$
  - Compute the integral $\int \; p_\theta(x|z)p_\theta(z)dz$
  - VAEs offer a solution to both problems
- Intractability
  - For complex distributions computing $\mathrm{p(x)} = \int \; p_\theta(x|z)p_\theta(z)dz$ is intractable
  - Intractability of $p(x)$, is related to the intractability of the posterior $p_\theta(z|x)$: $p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)}$
  - VAE propose to approximate $p_\theta(z|x)$ and then $p_\theta(x)$ using simplified assumptions through a Variational approximation

VAEs description - Training

- Encoder – Approximate posterior
  - In order to turn the intractable posterior and learning problem into tractable problems, VAE introduce a parametric inference model $q_\phi(z|x)$
    - ϕ are the variational parameters
    - They are optimized s. t. $q_\phi(z|x) \approx p_\theta(z|x)$
    - $q_\phi(z|x)$ is parameterized using an encoder NN
      - e.g. $q_\phi(z|x) = \mathcal{N}(\mu, diag(\sigma))$ with $(\mu, \sigma) = f_\phi(x)$ and $f_\phi(x)$ a NN
        - ϕ are then the weights and bias of the NN encoder

VAE- Description - Training
Evidence Lower Bound

- Log likelihood for data point x:
  - $\log p_\theta(x) = D_{KL}(q_\phi(z|x)||p_\theta(z|x)) + V_L(\theta, \phi; x)$
    - $D_{KL}(.||.) \geq 0$, then $V_L(\theta, \phi; x)$ is a lower bound of $\log p_\theta(x)$
  - Maximizing $\log p_\theta(x)$ is equivalent to maximizing $V_L(\theta, \phi; x)$ (and minimizing $\mathrm{D}_{KL}(q_\phi(z|x)||p_\theta(z|x))$
    - In order to maximize $\log p_\theta(x)$, we will maximize $V_L(\theta, \phi; x)$
    - This will approximately maximize the likelihood and reduce the gap between $q_\phi((z|x)||p_\theta)$ and $p_\theta(z|x)$

VAE- Description - Training
Evidence Lower Bound

- Variational lower bound (ELBO: Evidence Lower Bound):
  - $V_L(\theta, \phi; x) = -D_{KL}(q_\phi(z|x)||p(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$

- Because each representation z is associated to a unique x, the loss likelihood can be decomposed for each data point
  - The global log likelihood is then the summation of these individual losses
  - $V_L(\theta, \phi) = \sum_i V_L(\theta, \phi; x^i)$

VAE Description - Training
Derivation of the loss function

- $\log p_\theta(x) = \mathrm{D}_{KL}(q_\phi(z|x)||p_\theta(z|x)) + V_L(\theta, \phi; x)$
  - $\log p_\theta(x) = \int_z (\log p(x)) q(z|x)\, dz$ (since $\int_z q(z|x)\, dz = 1$)
  - $\log p_\theta(x) = \int_z (\log \frac{p(x,z)}{p(z|x)}) q(z|x)\, dz$
  - $\log p_\theta(x) = \int_z (\log \frac{p(x,z)}{q(z|x)} \frac{q(z|x)}{p(z|x)}) q(z|x)\, dz$
  - $\log p_\theta(x) = \int_z (\log \frac{p(x,z)}{q(z|x)}) q(z|x)\, dz + \int_z (\log \frac{q(z|x)}{p(z|x)}) q(z|x)\, dz$
  - $\log p_\theta(x) = E_{q(z|x)}[\log \mathrm{p(x,z)} - \log q(z|x)] + D_{KL}(q(z|x)||p(z|x))$
  - $\log p_\theta(x) = V_L(\theta, \phi; x) + \mathrm{D}_{KL}(q_\phi(z|x)||p_\theta(z|x))$

VAE Description - Training
Derivation of the loss function

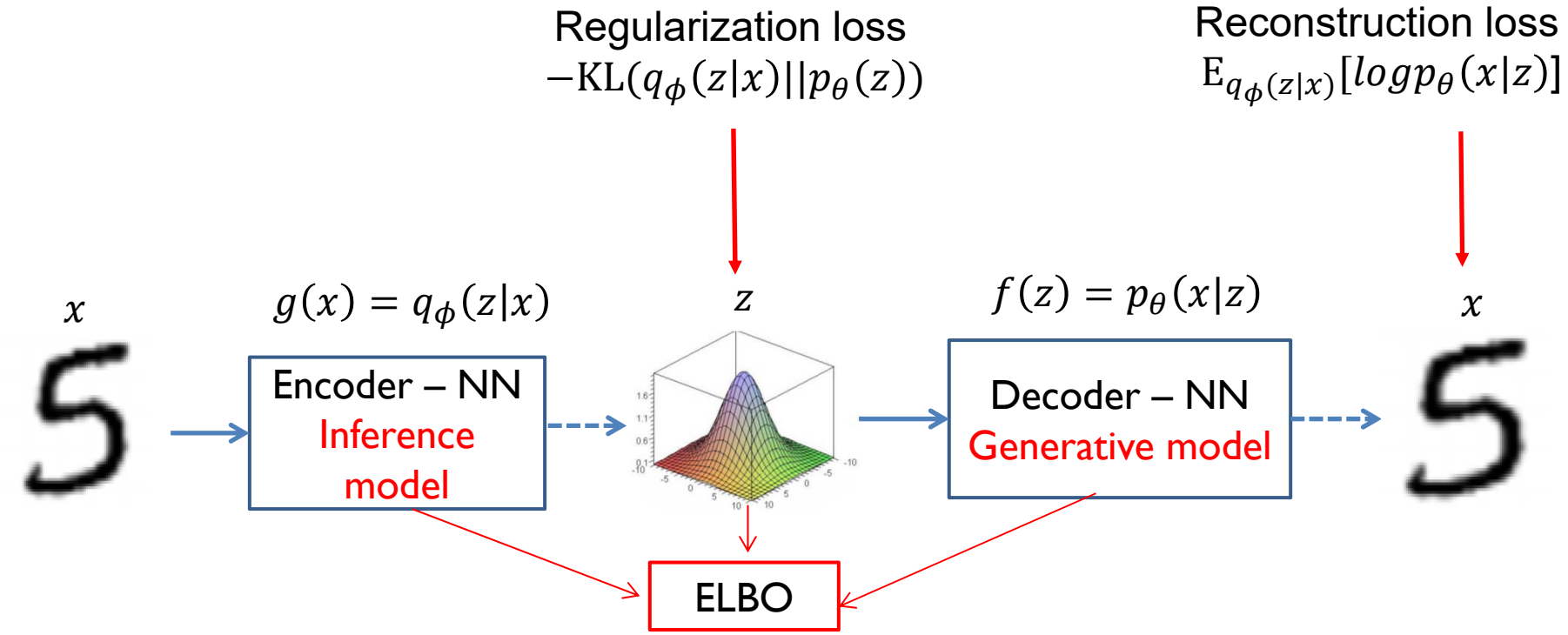
- $V_L(\theta, \phi; x) = -D_{KL}(q_\phi(z|x)||p_z(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$
  - $V_L(\theta, \phi; x) = \mathrm{E}_{q_\phi(z|x)}[log p_\theta(x, z) - \log q_\phi(z|x)]$
  - $V_L(\theta, \phi; x) = \mathrm{E}_{q_\phi(z|x)}[log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x)]$
  - $V_L(\theta, \phi; x) = -\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x)||p_\theta(z)) + \mathrm{E}_{q_\phi(z|x)}[log p_\theta(x|z)]$

VAE
ELBO – summary

- Variational lower bound:
  - $V_L(\theta, \phi; x) = -D_{KL}(q_\phi(z|x)||p(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$
  - Remarks
    - $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$ is a **reconstruction** term
      - Measures how well the datum x can be reconstructed from latent representation z
    - $D_{KL}(q_\phi(z|x)||p(z))$ is a **regularization** term:
      - Forces the learned distribution $q_\phi(z|x)$ to stay close to the prior $p(z)$
      - $p(z)$ has usually a simple form e.g. $\mathcal{N}(0, I)$, then $q_\phi(z|x)$ is also forced to remain simple
  - This form provides a link with a NN implementation
    - The generative $p_\theta(x|z)$ and inference $q_\phi(z|x)$ modules are implemented by NNs
    - They will be trained to maximize the reconstruction error for each (z, x): $E_{q_\phi(z|x)}[\log p_\theta(x|z)]$ term
    - The inference module $q_\phi(z|x)$ will be constrained to remain close to the prior $p(z)$: $-\mathrm{D}_{\mathrm{KL}}(q_\phi(z|x)||p_\theta(z)) \approx 0$

VAE
Loss - summary

- ELBO loss in the NN model



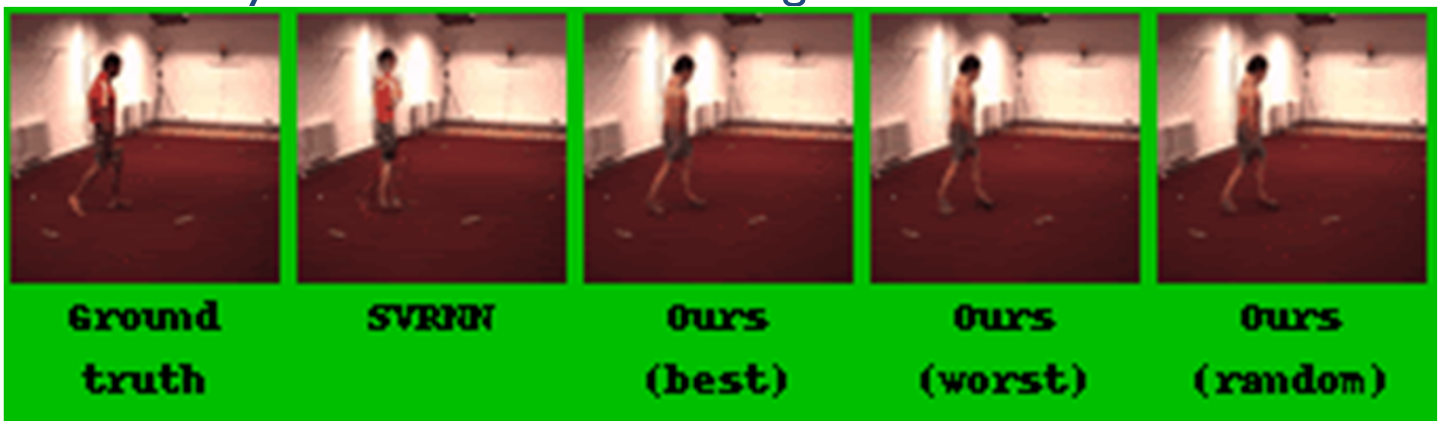
- Training performed via Stochastic gradient
  - ELBO allows for the simultanous training of the ϕ and θ parameters
  - This requires an analytical expression for the loss functions and for gradient computations
    - ⇢ Sampling
    - → deterministic

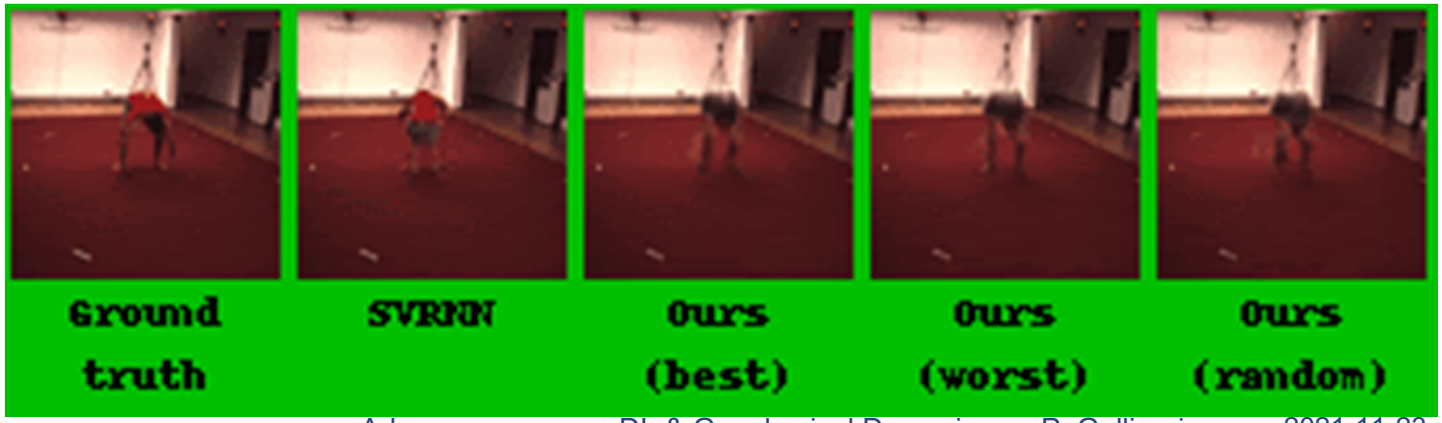
VAE- reparametrization trick

- Training with stochastic units: reparametrization trick
  - Not possible to propagate the gradient through stochastic units (the zs and xs are generated via sampling)
  - Solution for continuous distributions and differentiable encoder-decoder
    - Parametrize z as a deterministic transformation of a random variable ϵ:
      - If $z \sim \mathcal{N}(\mu, \sigma)$, it can be reparameterized by $z = \mu + \sigma \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0,1)$, with $\odot$ pointwise multiplication (μ, σ are vectors here)
      - For the NN implementation we have: $z = \mu_z(x) + \sigma_z(x) \odot \epsilon_z$
    - This will allow the derivative to « pass » through the z

Example: Stochastic video prediction (Franceschi et al. 2020)

- Human3.6M
  - SOTA video prediction rely on stochastic models
  - Stochasticity often introduced through conditional VAEs





VAEs – Testing the model

- Once trained, the model could be used to generate examples using only the decoder
  - - sample $z \sim \mathcal{N}(0, \mathrm{I})$, compute $g_\theta(z)$, sample to generate a datum

VAE summary

- Work with non invertible and non smooth g_θ
  - Easily handles discrete distributions
- Training relies on ELBO
- No saddle point pb like for GANs
- Often used to learn relevant priors for task specific models
  - E.g. stochastic video prediction

Normalizing Flows

Normalizing Flows
Obectives

- Normalizing Flows are a family of generative models which produces tractable distributions where both sampling and density evaluation can be **exact** (this is not the case for GANs and VAEs)
  - They can be trained by maximum likelihood
- They operate by **pushing** a simple **base density** through a **series of invertible and differentiable transformations** to produce a richer and **more complex** distribution
  - This is achieved through the change of variable theorem for densities

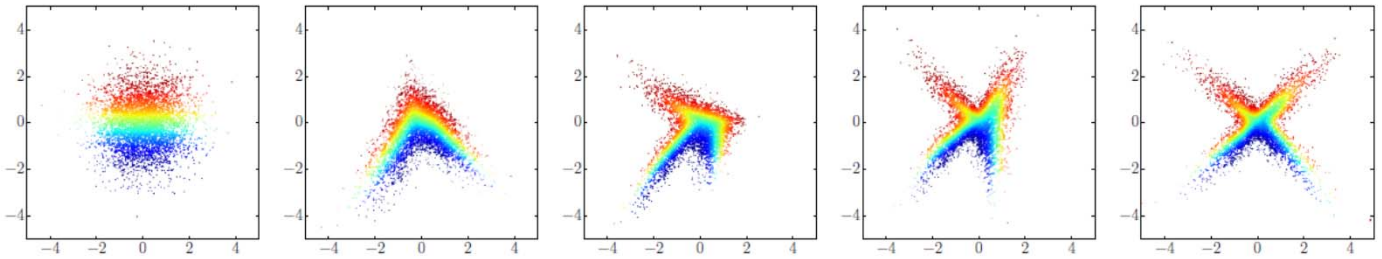


Figure 1: Example of a 4-step flow transforming samples from a standard-normal base density to a cross-shaped target density.

Fig. Papamakarios et al. 2019

Normalizing Flows
Prerequisite - Change of variable formula

- Notations
  - $x \in X$ observed variable, $x \in R^n$, $x \sim p_x$
  - $z \in Z$ latent variable, $z \in R^n$, $z \sim p_z$
  - $g: Z \to X$ invertible continuous function (bijection), $f = g^{-1}$ is the inverse of g
  - Both g and f are differentiable, i.e. are diffeomorphisms
    - Diffeomorphism
      - $g: U \to V$ with U and V open sets of R^n is a **diffeomorphism** if $1)$ g is bijective, $2)$ g is differentiable on U, $3)$ its inverse is differentiable on V
  - Under these conditions, the density of x is well defined and can be obtained by a change of variables

Normalizing Flows
Prerequisite - Change of variable formula

- Change of variable formula
  - $p_x(x) = p_z(z) \left|\det\left(\frac{\partial g(z)}{\partial z}\right)\right|^{-1}$
    - We denote $J_g(z) = \frac{\partial g(z)}{\partial z}$ the Jacobian of g at z
    - $p_x(x)$ is called the **pushforward of the density p_z by g**, and denoted $g_{\#}p_z$
  - Equivalently
  - $p_x(x) = p_z(f(x)) \left|\det\left(\frac{\partial f(x)}{\partial x}\right)\right|$
    - We denote $J_f(x) = \frac{\partial f(x)}{\partial x}$ the Jacobian of f at x
- Constructing a Flow based model, is made by implementing f or g via a neural network, taking p_z to be a simple density (e.g. multivariate Gaussian)

Normalizing Flows
Prerequisite – Jacobian – Determinants

- Jacobian
  - $J_g(z) = \frac{\partial g}{\partial z} = \begin{pmatrix} \frac{\partial g_1}{\partial z_1} & \cdots & \frac{\partial g_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial z_1} & \cdots & \frac{\partial g_n}{\partial z_n} \end{pmatrix}$
  - The determinant of a triangular matrix is the product of the diagonal terms

- Compositionality
  - Diffeomorphisms are composable
  - Let g_1 and g_2 be two diffeomophisms, and $g_2 \circ g_1$ their composition, then:
    - $(\mathrm{g}_2 \circ g_1)^{-1} = g_1^{-1} \circ g_2^{-1}$
    - $\det\left(\frac{\partial g_2 \circ g_1(z)}{\partial z}\right) = \det\left(\frac{\partial g_2 \circ g_1(z)}{\partial g_1(z)}\right) . \det\left(\frac{\partial g_1(z)}{\partial z}\right)$

Normalizing Flows

- Compositionality properties mean that complex transformations can be built from simple ones without loosing the properties of inversibility and differentiability
  - In practice, complex transformations g, can be built by composing simpler transformations $g = g_K \circ \cdots \circ g_1$, with each g_k transforming z_{k-1} to z_k, assuming $z_0 = z$ and $z_K = x$
  - The term **flow** refers to the trajectory that a collection of samples from $p_z(z)$ follow as they are transformed by the sequence of transformations $g_1, \ldots, g_K$
  - The term **normalizing** refers to the fact that the inverse flow $f = f_1 \circ \cdots \circ f_K$ takes a collection of samples from $p_x(x)$ and transforms them into a collection from a prescribed density $p_z(z)$, in a sense normalizing them

Normalizing Flows
Density estimation and sampling

- A flow based model provides two operations:
  - 1. sampling via $x = g(z)$, where $z \sim p_z(z)$
  - 2. density estimation via $p_x(x) = p_z(f(x)) \left|\det\left(\frac{\partial f(x)}{\partial x}\right)\right|$

g: sampling direction

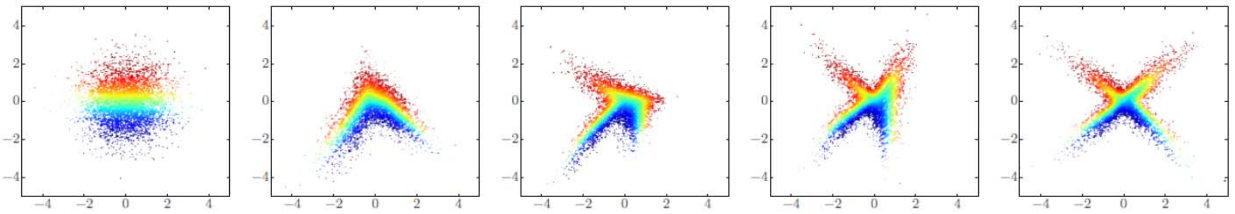


Figure 1: Example of a 4-step flow transforming samples from a standard-normal base density to a cross-shaped target density.

f: density estimation direction

Fig. Papamakarios et al. 2019

Normalizing Flows
Density estimation and sampling – Training the flow model

- Let θ and ϕ be respectively the parameters of flow g and base density p_z, and $\Theta = (\theta, \phi)$
- Given observed data $D = \{x^i\}_{i=1\ldots N}$ the parameters can be estimated by maximum likelihood
- The data likelihood is
  - $\mathrm{L}(\Theta) = \log p(D;\Theta) = \sum_{i=1}^{N} \log p_x(x^i;\Theta)$
  - $\mathrm{L}(\Theta) = \log p(D;\Theta) = \sum_{i=1}^{N} \log p_z(f(x^i;\theta);\phi) + \log \left|\det\left(\frac{\partial f(x^i;\theta)}{\partial x}\right)\right|$
    - The first term is the likelihood of the sample under the base measure
    - The second term accounts for the change of volume induced by the transformation
- The parameters can be estimated iteratively with a stochastic gradient algorithm. The gradients of $L(\Theta)$ w.r.t. parameters θ and ϕ

Normalizing Flows
Constructing flows

- Finite composition
  - The flow is described as a finite composition of transformations $f = f_K \circ \cdots \circ f_1$
  - Examples :
    - Coupling layers, Auto-regressive flows, Residual Flows
- Continuous time composition
  - The flow is described as a continuous time function, i.e. its dynamics is described by an Ordinary Differential Equation (ODE)

Normalizing Flows
Constructing flows – finite composition

- The flow is described as a finite composition of transformations
  $g = g_K \circ \cdots \circ g_1$
  - Each g_i implements a simple transformation $g_i: R^n \to R^n$, with a tractable inverse and Jacobian determinant. Let us denote $z_0 = z$ and $z_K = x$
- The forward evaluation is:
  - $z_k = g_k(z_{k-1})$
- The determinant of the jacobian is:
  - $\log \left|\frac{\partial g(z)}{\partial z}\right| = \sum_{k=1}^{K} \log \left|\frac{\partial g_k(z_{k-1})}{\partial z_{k-1}}\right| = \sum_{k=1}^{K} \log \left|\frac{\partial z_k}{\partial z_{k-1}}\right|$
- Same thing for the inverse transformation
- $g_k$or f_k will be implemented with a NN,
- In each case, one must ensure that the NN is invertible and has a tractable determinant
  - Depending on the application, one will implement either $g_k$or f_k

Normalizing Flows
Constructing flows – finite composition – **Coupling layers**

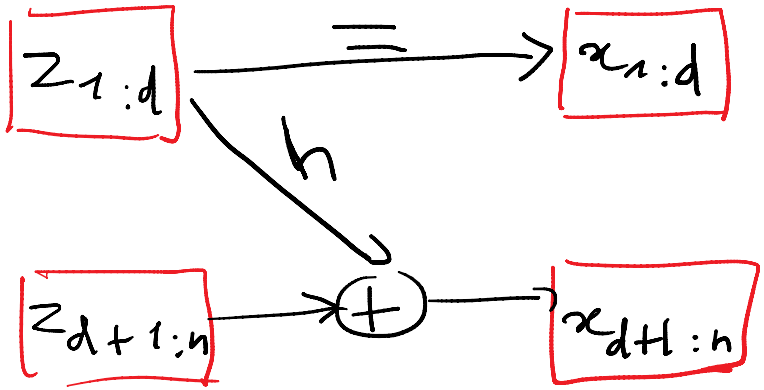
- We describe different possibilities for implementing the g_ks
  - For notation simplification, we consider a single tranformation g, and the inputs/ outputs are respectively denoted z and x.
- **Coupling layers** make use of transformations which are easily invertible and for which the Jacobian determinant is easily computable.

Normalizing Flows
Constructing flows – finite composition - Coupling layers – NICE (Dinh 2015)

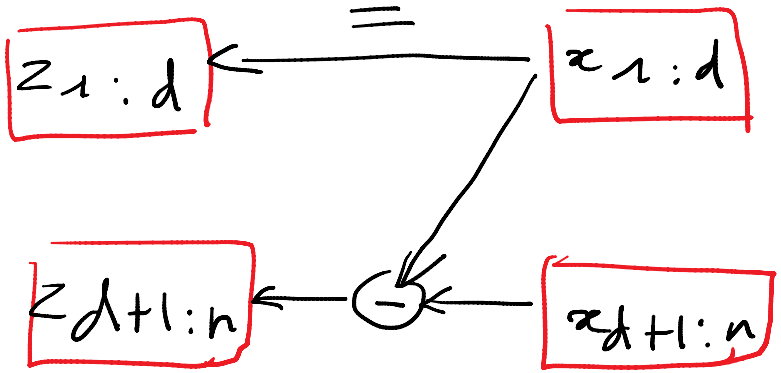
- Coupling layers were introduced in (Dinh 2015)
  - Forward transformation are implemented as **additive** transformations
    - $x_{1:d} = z_{1:d}$ (identity)
    - $x_{d+1:n} = z_{d+1:n} + \mathrm{h}(z_{1:d})$ with h a NN with d input units and $n - d$ output units
  - Inverse transformation can be easily computed
    - $z_{1:d} = x_{1:d}$ (identity)
    - $z_{d+1:n} = x_{d+1:n} - \mathrm{h}(x_{1:d})$
    - Computing the inverse is as simple as computing the forward transformation
  - Jacobian of the forward mapping
    - $J = \frac{\partial x}{\partial z} = \begin{pmatrix} I_d & O \\ \frac{\partial x_{d+1:n}}{\partial z_1{:}d} & I_{n-d} \end{pmatrix}$
    - $\det(J) = 1$, i.e. this is a volume preserving transformation

Normalizing Flows
Constructing flows – finite composition - Coupling layers – NICE (Dinh 2015)

- Forward

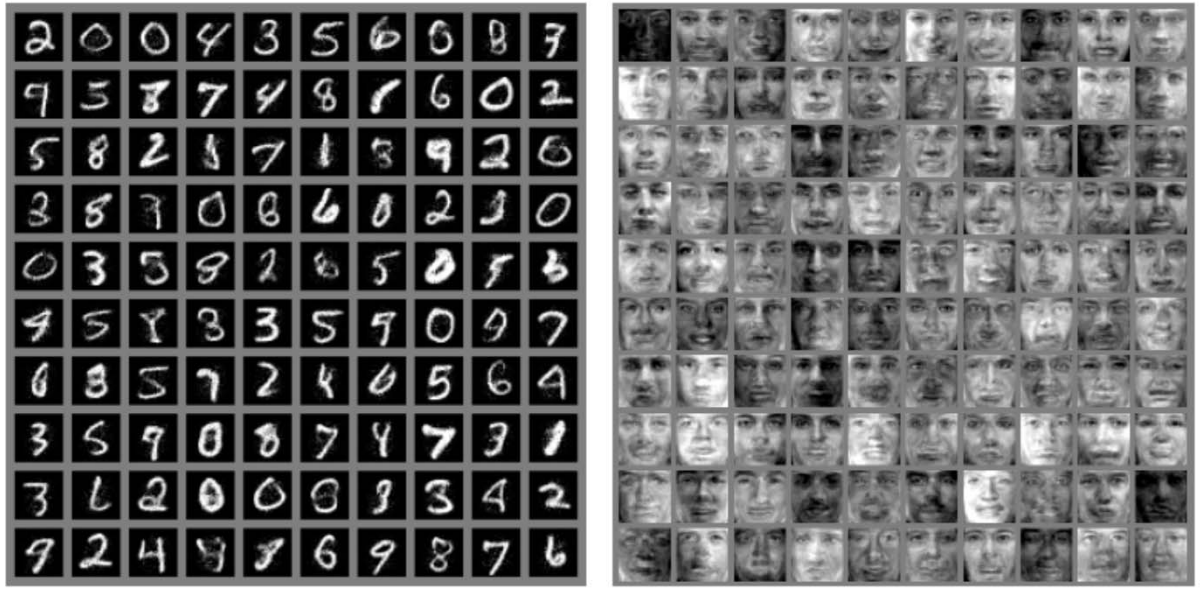


- Reverse



Normalizing Flows
Constructing flows – finite composition - Coupling layers – NICE (Dinh 2015)

- Samples generated via NICE
  - $z \sim p_z(z)$ and then $x = f_\theta(z)$
  - $p_z(z)$ is a logistic distribution for MNIST and Gaussian for TFD



(a) Model trained on MNIST (b) Model trained on TFD

- They also evaluate the quality of the model likelihood on the data

Normalizing flows

- Several families of finite normalizing flows
- Alegbraic inversible construction
  - Efficient g and f
    - NICE (Dinh et al. 2014)
    - Real non-volume preserving flows (Dinh et al. 2016) – extends NICE
  - Efficient g, not f
    - Planar and radial flows (Rezende et al. 2015)
    - Inverse autoregressive flows (Kingma et al. 2016)
  - Efficient f, not g
    - Masked autoregressive flows (Papamakarios et al. 2017)
- Numerical inverse construction
  - Revnet - iRevnet

Continuous normalizing flows (Grathwohl et al 2019)

- Formulated as an ordinary differential equation (ODE) problem
  - i.e. continuous time instead of finite function composition
  - The generator is defined as
    - $g_\theta(z) = y(T), T > 0$ being a terminal time
    - $y: [0, T] \to R^n$ satisfies the initial value problem (ODE)
      - $\frac{\partial y(t)}{\partial t} = h_\theta(y(t), t)$ with $y(0) = z$
      - $h_\theta: R^n \times R \to R^n$ is a NN parameterized by θ
      - The mapping h_θ is invertible and one may define the inverse of g_θ

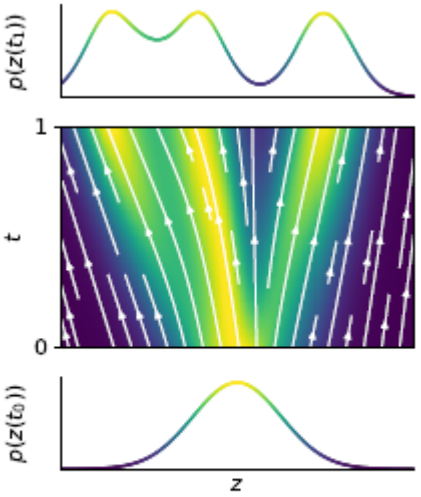


Figure 1: FFJORD transforms a simple base distribution at t_0 into the target distribution at t_1 by integrating over learned continuous dynamics.

Fig. Grathwohl et al. 2019

Normalizing Flows summary

- Trained by max likelihood
- Can be used for sampling and density estimation
- g diffeomorphic: input/ output should be the same dimension
- Assume smoothness of the generator
- Often used as a complement yo other generative models in order to get better posteriors

References: papers used as illustrations for the presentation

- Generative models
  - L. Ruthotto and E. Haber. An introduction to deep generative modeling, 2021. Course notes
- GANs
  - Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein Generative Adversarial Networks. In Proceedings of The 34th International Conference on Machine Learning (pp. 1–32). Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In ICCV (pp. 2223–2232).
  - Chen M. Denoyer L., Artieres T. Multi-view Generative Adversarial Networks without supervision, 2017 , https://arxiv.org/abs/1711.00305.
  - de Bezenac, E., Ayed, I. and Gallinari, P. 2021. CycleGan through the lens of (Dynamical) Optimal Transport, ECML 2021
  - Goodfellow I, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio , Generative adversarial nets, NIPS 2014, 2672-2680
  - Goodfellow, I. NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv:1701.00160, Dec. 2016.
  - Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A.P., Tejani, A., Totz, J., Wang, Z. and others 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *Cvpr*. 2, 3 (2017), 4.
  - Leinonen, J., Nerini, D. and Berne, A. 2020. Stochastic Super-Resolution for Downscaling Time-Evolving Atmospheric Fields With a Generative Adversarial Network. IEEE Transactions on Geoscience and Remote Sensing. 59, 9 (2020), 7211–7223.
  - Mirza, M., & Osindero, S. (2014). Conditional Generative Adversarial Nets. In arxiv.org/abs/1411.1784.
  - Radford, Luke Metz, Soumith Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2016, http://arxiv.org/abs/1511.06434
  - Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., Prudden, R., Mandhane, A., Clark, A., Brock, A., Simonyan, K., Hadsell, R., Robinson, N., Clancy, E., Arribas, A. and Mohamed, S. 2021. Skilful precipitation nowcasting using deep generative models of radar. *Nature*. 597, 7878 (2021), 672–677.
  - Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. and Lee, H. 2016. Generative Adversarial Text to Image Synthesis. Icml (2016), 1060–1069.
  - Xie, Y., Franz, E., Chu, M. and Thuerey, N. 2018. tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow. *Siggraph* (2018).

References: papers used as illustrations for the presentation

- VAEs
  - Kingma D.P., Welling, M., et al. An introduction to variational autoencoders, Foundations and trends in Machine Learning, 12 (4), 2019
  - Franceschi, J.-Y., Delasalles, E., Chen, M., Lamprier, S. and Gallinari, P. 2020. Stochastic Latent Residual Video Prediction. *ICML* (2020).

- Normalizing Flows
  - Behrmann, J., Grathwohl, W., Chen, R.T.Q., Duvenaud, D., and Jacobsen, J.-H. 2019. Invertible Residual Networks. ICML.
  - Chen, R.T.Q., Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. 2019. Residual Flows for Invertible Generative Modeling. 1–21.
  - Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. arXiv:1605.08803, May 2016.
  - Dinh, L., Krueger, D., and Bengio, Y. 2015. NICE: Non-linear Independent Components Estimation. ICLR Wshop.
  - Dinh, L., Sohm-Dickstein, J., and Bengio, S. 2017. Density Estimation using Real NVP. ICLR.
  - Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In International Conference on Learning Representations, 2018.
  - Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. arXiv:1606.04934, June 2016.
  - Kobyzev, I., Prince, S., and Brubaker, M.A. 2019. Normalizing Flows: Introduction and Ideas. 1–35.
  - Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., and Lakshminarayanan, B. 2019. Normalizing Flows for Probabilistic Modeling and Inference. 1–60.
  - Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In Advances in Neural Information Processing Systems, pages 2338{2347, 2017.
  - Rezende and S. Mohamed. Variational inference with normalizing flows. In International Conference on Machine Learning, pages 1530-1538, 2015.