3rd ML-INFN Hackathon: *Advanced Level*

INTRODUCTION TO GENERATIVE MODELS

GAN, VAE and Flow-based models

Matteo Barbetti (INFN-Firenze, University of Firenze)

Francesco Vaselli (INFN-Pisa, Scuola Normale Superiore)

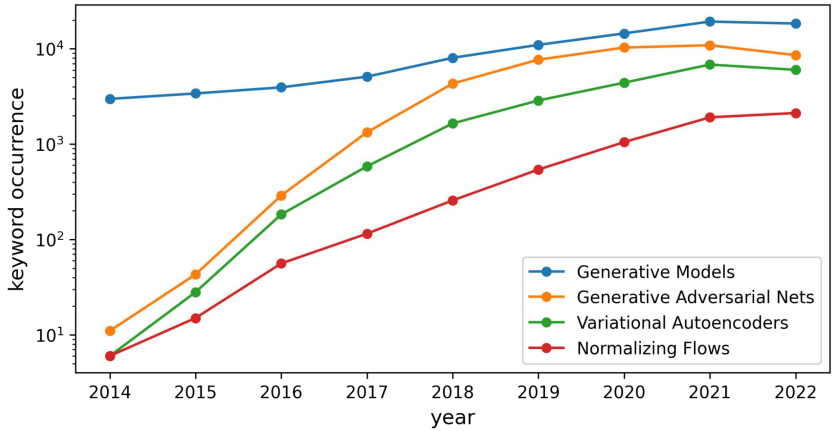
OUTLINE

1. **What is generative modeling?**
   - *Overview of state-of-the-art algorithms*
2. **Generative Adversarial Networks**
   - *Definition, drawbacks and improvements*
3. **Variational Autoencoders**
   - *Definition, probabilistic interpretation and known problems*
4. **Normalizing Flows**
   - *A family of methods for constructing flexible learnable probability distributions*
5. **Physics use-cases**
   - *How generative models are used in our field*

1. WHAT IS GENERATIVE MODELING?

Overview of state-of-the-art algorithms

A look at academic trends



Data drawn from Google Scholar using
Pold87/academic-keyword-occurrence

The interest of Scientific Community towards **Generative Models** has grown in recent years thanks to Deep Learning developments.

Usage and further improvement of this or the other algorithm are driven by how much it looks promising to cope a **specific task**.

Some applications

Generative modeling algorithms have been mainly developed for applications in **Computer Vision** to create, transform or improve a broad set of images composed of human faces, animals, landscapes, cartoons, sketches, photos and much more.

Generative models can be mainly used for:

- **Data augmentation**
  - Creation of new and never-seen instances according to the reference dataset
- **Super resolution**
  - Enhancing the resolution of an input image keeping its quality as high as possible
- **Inpainting**
  - Reconstruction of missing pixels in the input images keeping realism and consistency
- **Denoising**
  - Removing noise from input instances minimizing the loss of information
- **Translation**
  - Creation or transformation of images from a domain to another (i.e. text-to-image translation)

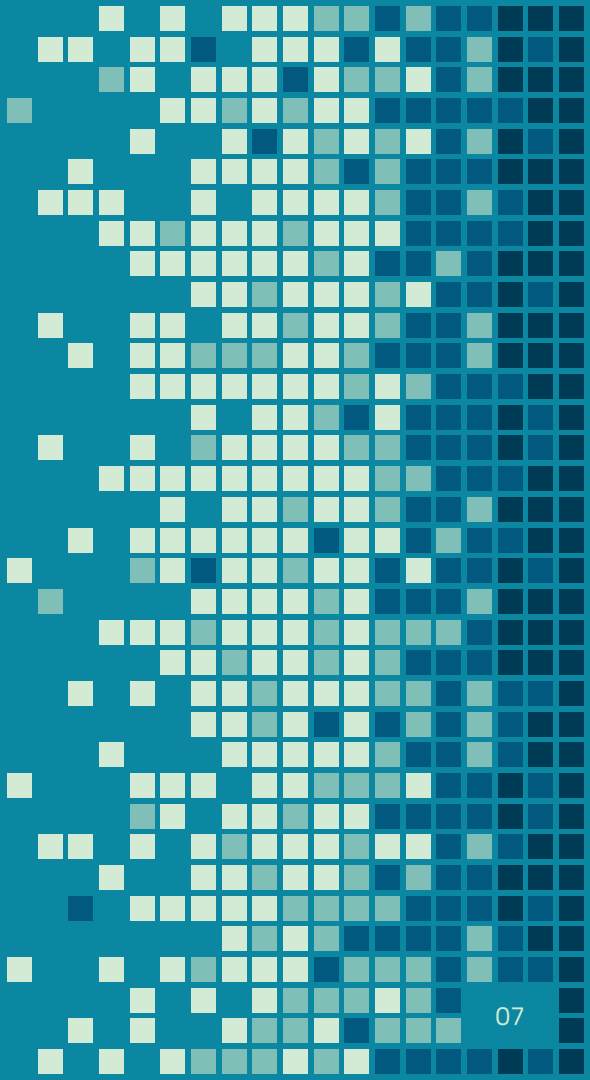
Text-to-image with DALL·E 2

"Teddy bears mixing sparkling chemicals as mad scientists as a 1990s Saturday morning cartoon"




Images taken from the OpenAI blog post

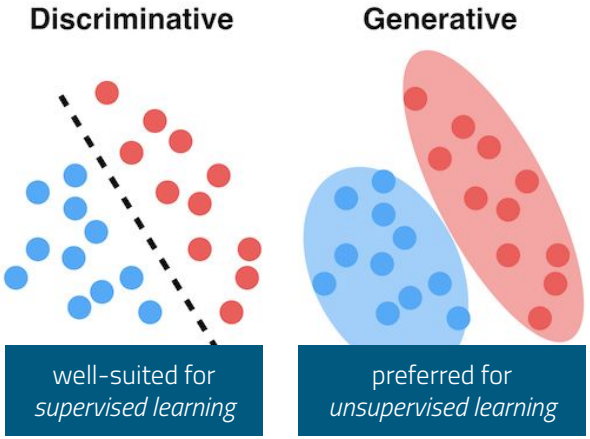
> *What I cannot create,*
> *I do not understand.*
>
> – Richard Feynman



A more general formulation

Let X and Y be respectively the set of data instances and the corresponding set of labels. Considering a Machine Learning classification task, the models that we can set up to face the problem can be broadly divided into two main categories:

- ***Discriminative model*** – model of the conditional probability of the target Y, given an observation x, symbolically P(Y|X=x);
- ***Generative model*** – model of the conditional probability of the observable X, given a target y, symbolically P(X|Y=y).



INPUT

$$\mathcal{D}_{\text{train}} = \{x_1, \ldots, x_N\} \quad \text{with } x_i \sim \mathrm{P}(x)$$

TARGET

$$d\left(\mathrm{P}(x), \hat{\mathrm{P}}_\theta(x)\right) \simeq 0$$

generative model

Machine-Learnt generative model

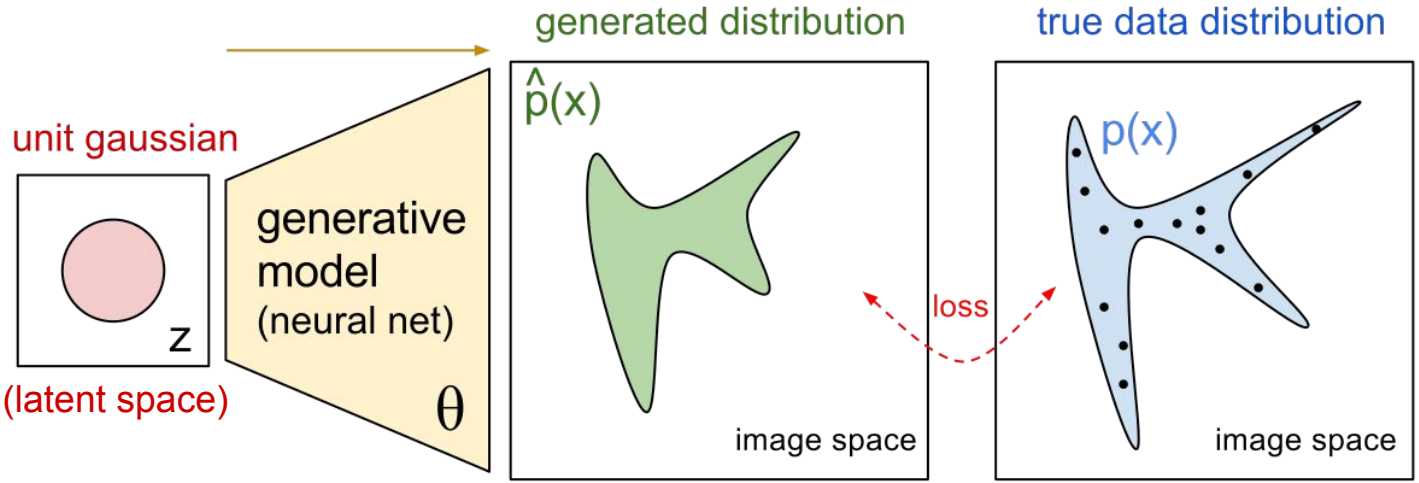


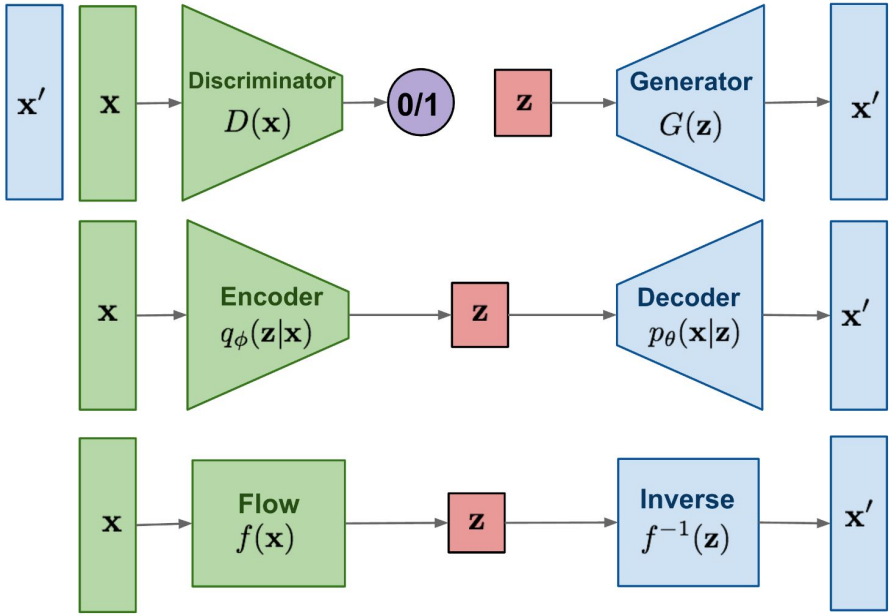
Figure stolen from the OpenAI blog post

Different types of generative models

GAN: Adversarial training

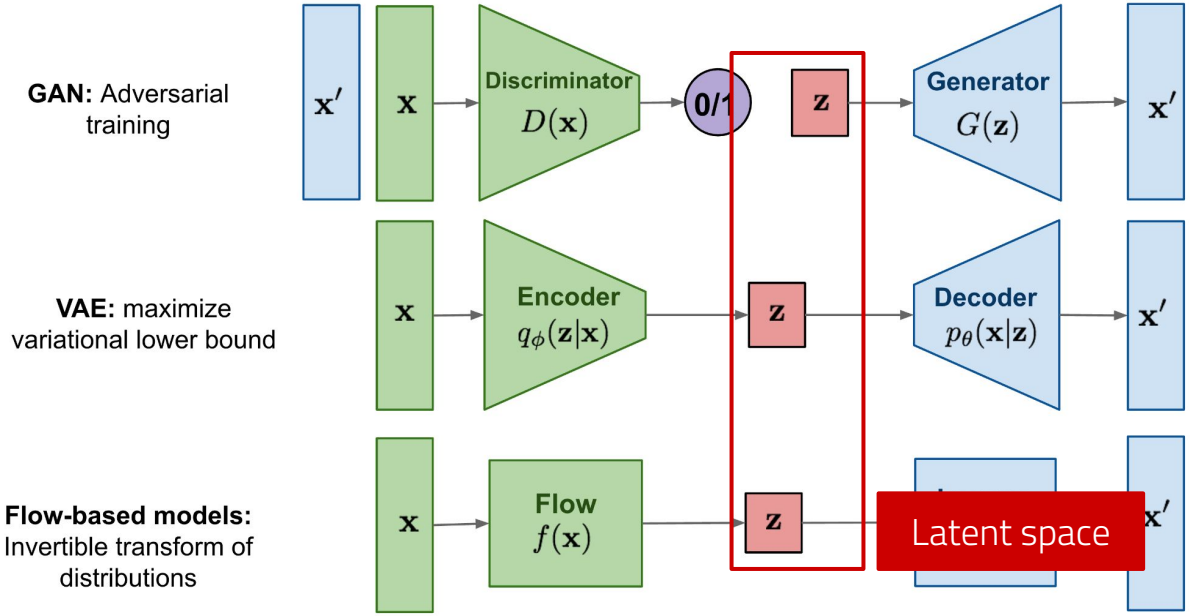
VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions



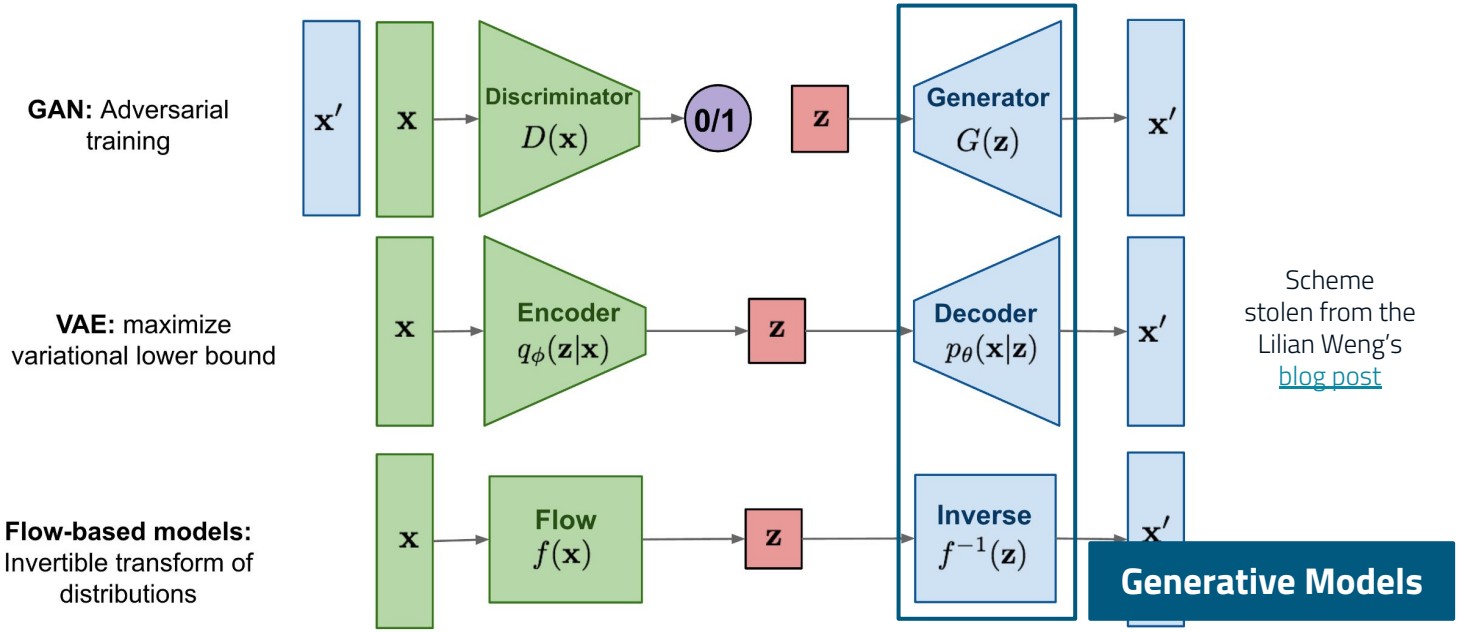
Scheme stolen from the Lilian Weng's blog post

Different types of generative models



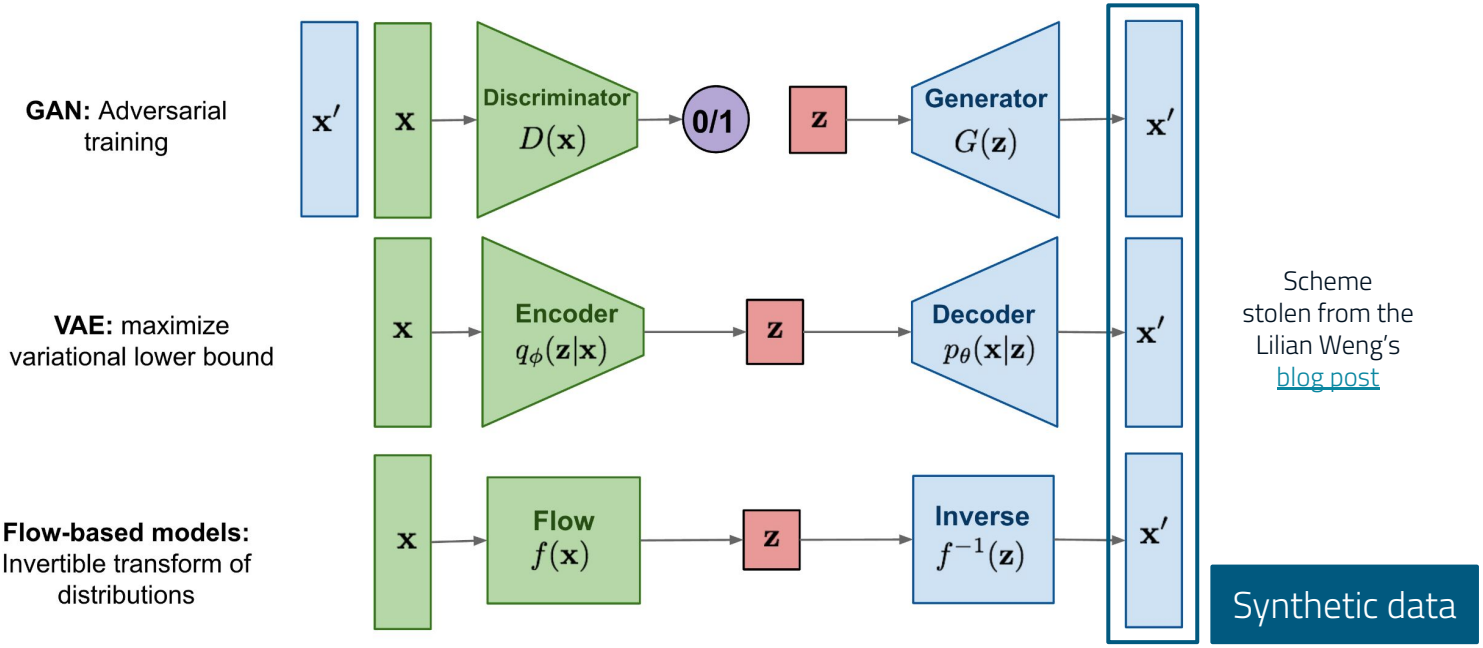
Scheme stolen from the Lilian Weng's [blog post]

Different types of generative models



Scheme stolen from the Lilian Weng's blog post

Different types of generative models



Scheme stolen from the Lilian Weng's blog post

Ready to use implementations

The main Machine Learning frameworks (Keras/TensorFlow/PyTorch) offer several tutorial to implement *vanilla* **generative modeling algorithms**, such as DCGAN or CVAE for the creation of digit images from the MNIST dataset.

Since they are quite recent algorithms, high-level solutions for Normalizing Flows are not yet available, unless through some **custom libraries** such as FlowTorch or PZFlow.

Some work is ongoing to provide a package for ready-to-use implementations of generative models for Physics applications:

- TFGenModels is based on TensorFlow 2 (work in progress);
- TorchGen is based on PyTorch (still to release).

TorchGen

TFGenModels

2. GENERATIVE ADVERSARIAL NETWORKS

Definition, drawbacks and improvements

Introduction to GANs

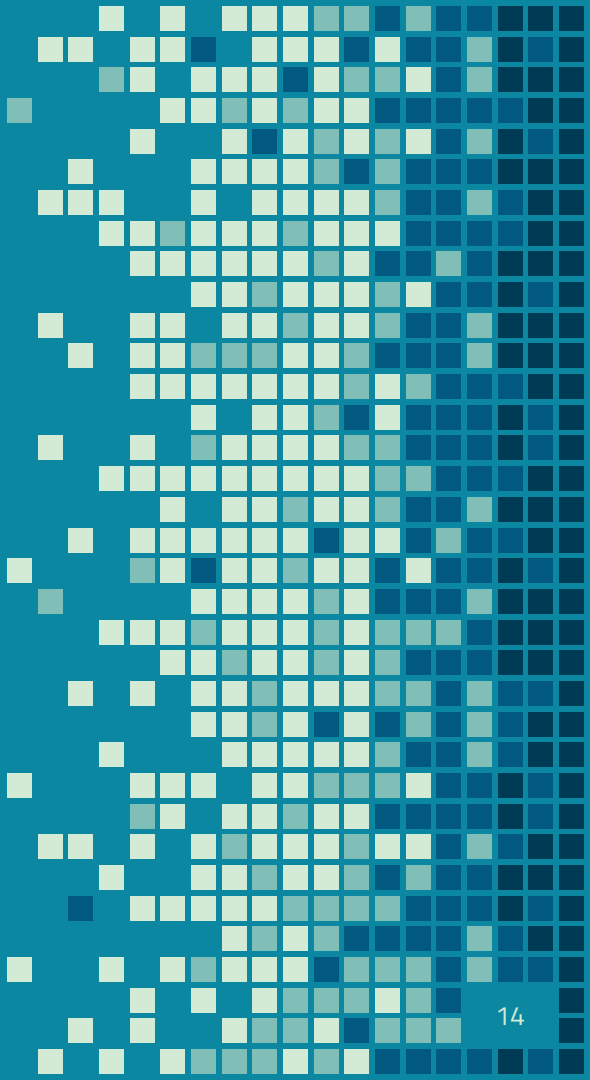
Generative Adversarial Networks (GAN) rely on the simultaneous training of two neural nets:

- the **discriminator network** (D) is trained by a classification task to separate the generator output from the reference dataset;
- the **generator network** (G) is trained by a simulation task to reproduce the reference dataset trying to fake the discriminator.

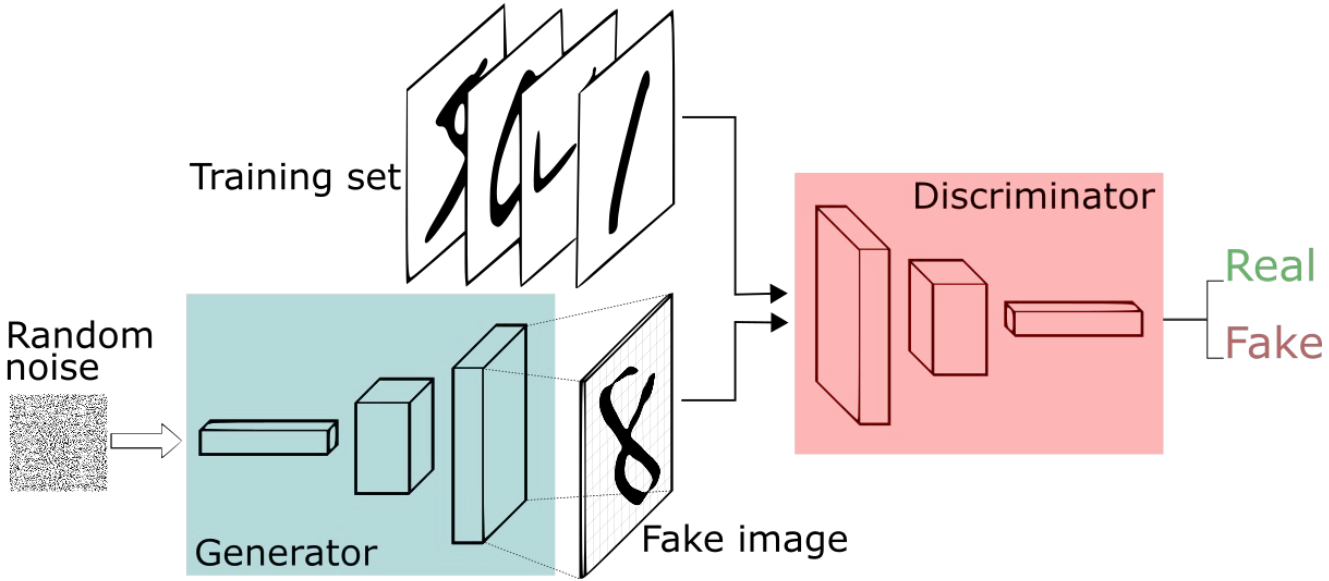
The discriminator goal is to **maximize** the separation between the generator output and the real data, while the generator, driven by the discriminator errors, aims to **minimize** the differences with the data.

This framework corresponds to a **minimax two-players game**

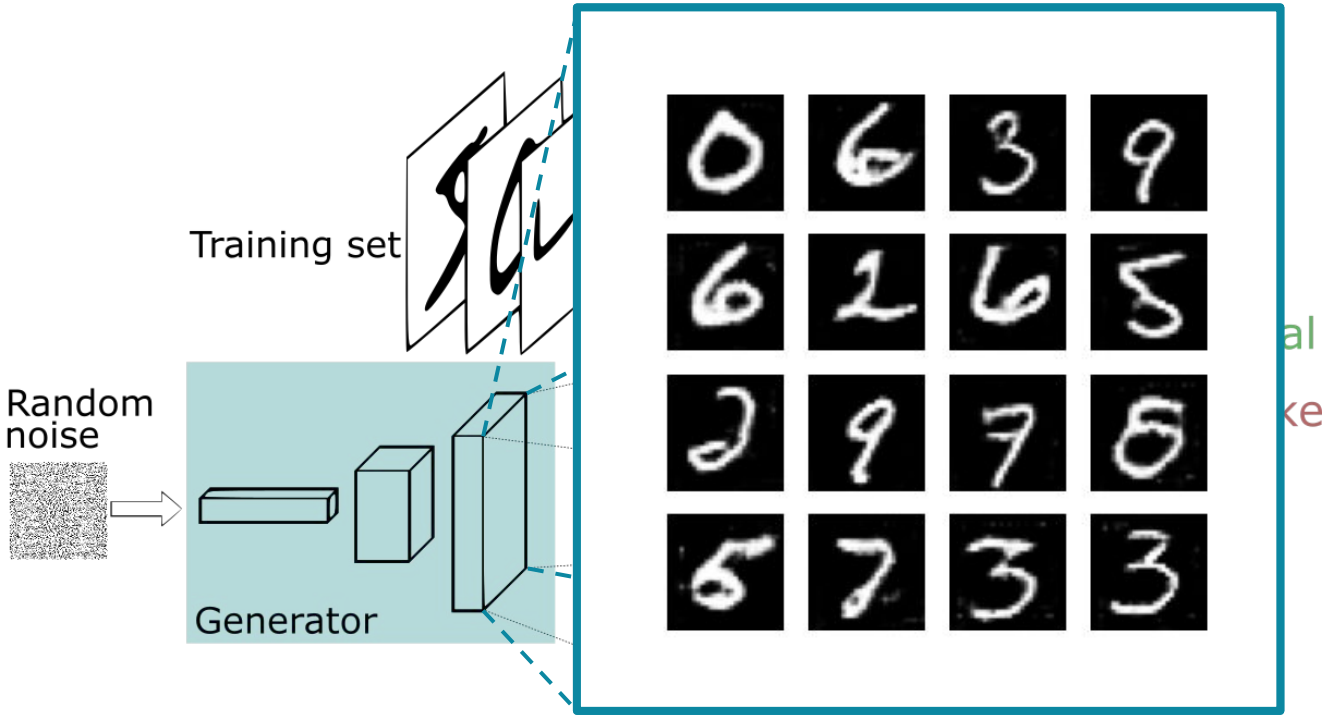
> *The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency.*
>
> – Ian J. Goodfellow *et al.*



Schematic representation



Schematic representation

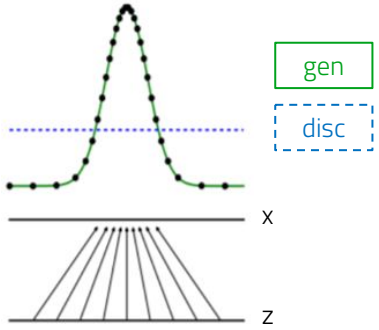


Mathematical definition

The original **loss function** proposed by Ian J. Goodfellow *et al.* is

$$\mathcal{L}_{\mathrm{GAN}}(D,G) = \mathbb{E}_{x\sim \mathrm{P}_{\mathrm{true}}}[\log D(x)] + \mathbb{E}_{z\sim \mathrm{P}_{\mathrm{z}}}[\log(1 - D(G(z)))]$$

where D outputs the probability that its input comes from the reference dataset. The generator goal is that the discriminator makes a mistake, resulting with the following **minimax game**:



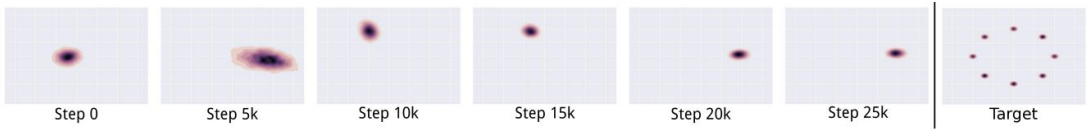
$$\min_G \max_D \mathcal{L}_{\mathrm{GAN}}(D,G)$$

Theoretically a unique solution exists, with G recovering the reference data distribution and D equal to ½ everywhere.

Unstable training dynamics

Traditional GAN systems suffer from many issues mainly due to an unstable training procedure:

- the generator may collapse producing only a single sample or a small family of very similar samples (problem named ***mode collapse***);
- the two players may oscillate during training rather than converging to the **equilibrium point**;
- if **imbalance** between the two agents occurs, then the system is incapable of learning at all.



Example of mode collapse taken from arXiv:1611.02163

All these drawbacks result from the ***vanishing gradient*** problem, namely the lack of information for the update of the generator parameters. This is due to the saturation of the discriminator that is so good in distinguishing the origin of the two samples that no errors remain to the generator to improve the synthetic space.

Stabilizing the training procedure

Stabilizing the training procedure relies on preventing the vanishing gradient problem, that can be achieved redefining the loss function to implement the ***Wasserstein distance*** to avoid the discriminator saturation.

$$\mathcal{L}_{\mathrm{WGAN}}(D, G) = \mathbb{E}_{x \sim \mathrm{P}_{\mathrm{true}}}[D(x)] - \mathbb{E}_{z \sim \mathrm{P}_{\mathrm{z}}}[D(G(z))]$$

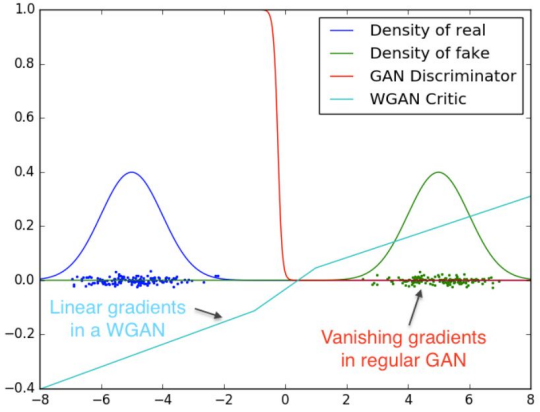


Figure taken from arXiv:1701.07875

with D parameterizing a **1-Lipschitz function**, that leads the minimax game to:

$$\min_{G} \max_{\|D\|_L \leq 1} \mathcal{L}_{\mathrm{WGAN}}(D, G)$$

Several training strategies have been developed to ensure the discriminator lipschitzianity, that can be done **indirectly** (i.e. with WGAN-GP) or **directly** (i.e. with WGAN-ALP).

References for GANs

- I. J. Goodfellow *et al.*, “Generative Adversarial Nets”, arXiv:1406.2661
- M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets”, arXiv:1411.1784
- A. Radford, L. Metz and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", arXiv:1511.06434
- L. Metz, “Unrolled Generative Adversarial Networks”, arXiv:1611.02163
- M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", arXiv:1701.04862
- M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein GAN", arXiv:1701.07875
- I. Gulrajani *et al.*, "Improved Training of Wasserstein GANs", arXiv:1704.00028
- M. G. Bellemare *et al.*, "The Cramer Distance as a Solution to Biased Wasserstein Gradients", arXiv:1705.10743
- D. Terjék, “Adversarial Lipschitz Regularization”, arXiv:1907.05681

3. VARIATIONAL AUTOENCODERS

Definition, probabilistic interpretation and known problems

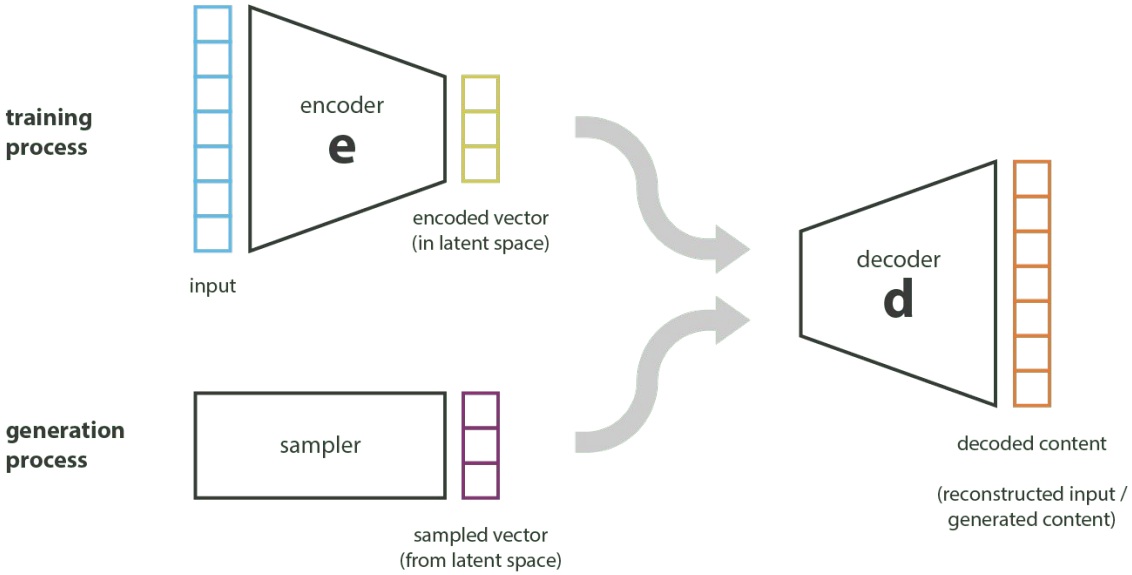
Introduction to VAEs

A ***Variational AutoEncoder*** (VAE) is an *autoencoder* (AE) whose training is regularized to avoid overfitting and ensure that the latent space has good properties that enable generative process. Also VAEs are based on the simultaneous training of two neural nets:

- the **encoder network** (e) defines a map from the reference space to the latent space with the goal of build a low-dimensional representation of the input;
- the **decoder network** (d) defines a map from the latent space to the synthetic space with the goal of maximize the matching between the output and the reference space.

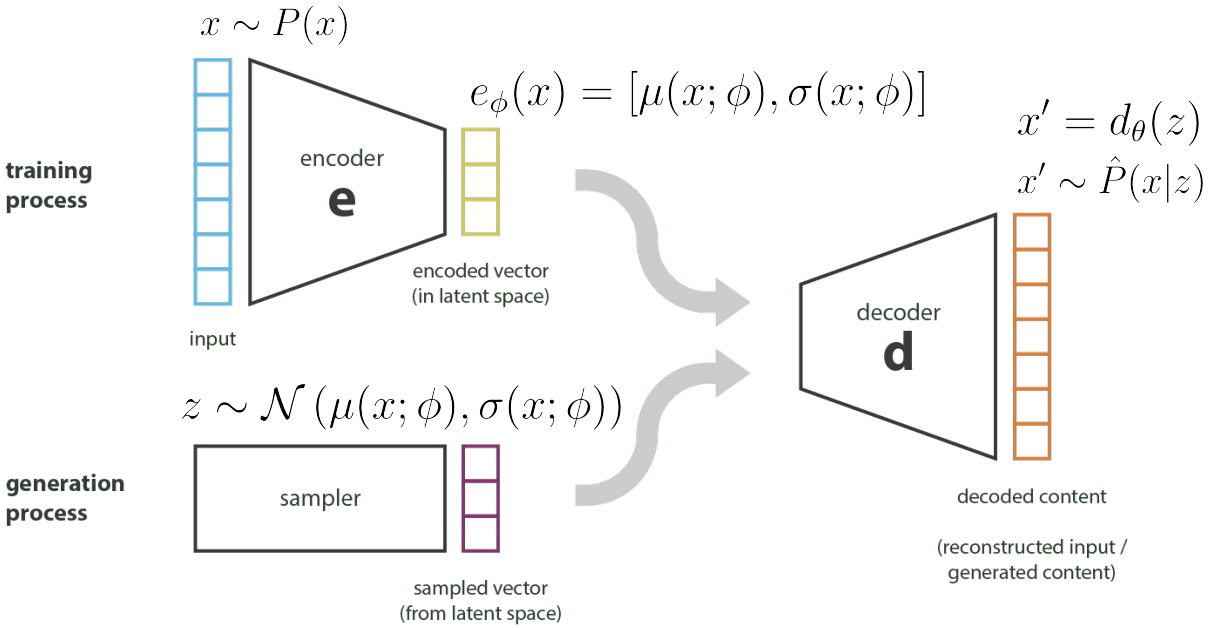
Contrary to GANs, VAEs training doesn't rely on an adversarial procedure and the system goal is to minimize the distance between encoded-decoded data and the initial data (**reconstruction error**). Moreover, contrary to AEs, VAEs encode an input as a **distribution** (with good properties) over the latent space rather than a single point to enable generative capabilities.

Schematic representation



Scheme stolen from the Joseph Rocca's [blog post]

Schematic representation



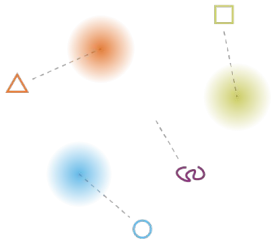
Scheme stolen from the Joseph Rocca's [blog post]

Mathematical details

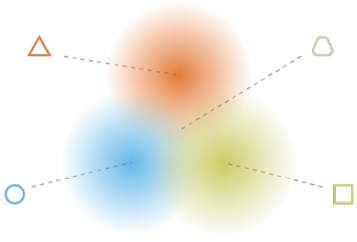
The VAE training is driven by the following **loss function**:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi) = \|x - d_\theta(z)\|^2 + D_{KL}\left(\mathcal{N}\left(\mu(x;\phi), \sigma(x;\phi)\right) \| \mathcal{N}(0, I)\right)$$

where the first term is the *reconstruction error*, while the second one is a *regularization term* that forces the distribution induced by the encoder to have good properties for the **generative process**.



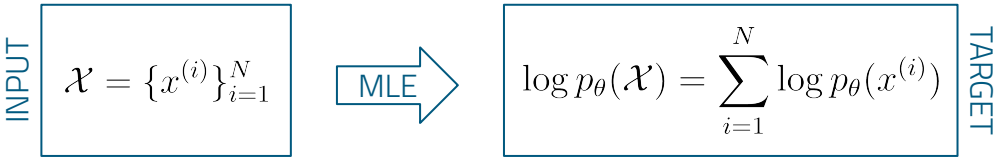
latent space
not regularized



latent space
regularized

Probabilistic interpretation

Let x be observed variables, z latent variables and let p(x,z) be the parametric model of their joint distribution.



Typically, this marginal log-likelihood is intractable to compute. Exploiting the **Bayes' Theorem**, one should try to tackle the MLE maximizing its lower bound, named ***Evidence Lower Bound*** (ELBO):

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] + D_{KL}\left(q_\phi(z|x)||p_\theta(z|x)\right) \\ &\geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x)||p(z)\right) \equiv -\mathcal{L}_{\text{VAE}}\end{aligned}$$

where q(z|x) is an *inference model* approximating the posterior p(z|x).

Known problems

Despite their strong probabilistic base, VAEs are and have been less used than GANs since the **limited capabilities** of the generative model provided. The major drawbacks of VAEs results from the approximation of the maximum likelihood previously mentioned:

- **blurry** or **fuzzy generated data** – q(z|x) maps multiple x to the same encoding z, making p(x|z) non-Gaussian for which the MSE within the loss produces some "average" of p(x|z);
- ***posterior collapse*** (also called ***optimization challenges*** or ***information preference***) – the decoder is so powerful to minimize "alone" the reconstruction error ignoring the latent space and making collapsing the posterior q(z|x) to the prior p(z), namely $q(z|x) \approx p(z)$ and $D_{KL}(q||p) \approx 0$.

Several strategies have been developed to cope these drawbacks, most of which based on a change of the loss function (i.e. VLAE or beta-VAE), or on a redesign of the VAE architecture (i.e. VQ-VAE), or even on a constraint over the variational models usable as approximators (i.e. delta-VAE).

Images stolen from the Enoch Kan's blog post

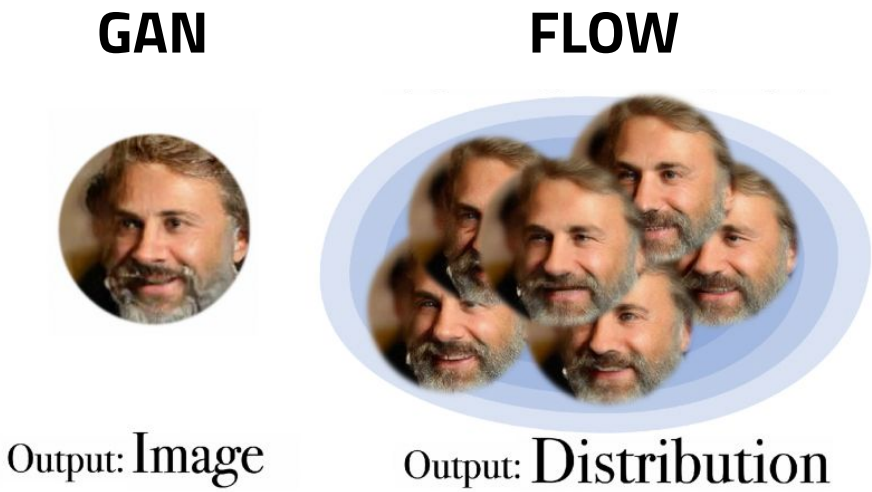
References for VAEs

- D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", arXiv:1312.6114
- D. J. Rezende, S. Mohamed and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models", arXiv:1401.4082
- X. Chen *et al.*, "Variational Lossy Autoencoder", arXiv:1611.02731
- S. Zhao, J. Song and S. Ermon, "Towards Deeper Understanding of Variational Autoencoding Models", arXiv:1702.08658
- I. Higgins *et al.*, "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework", ICLR 2017 proceeding
- A. van den Oord, O. Vinyals and K. Kavukcuoglu, "Neural Discrete Representation Learning", arXiv:1711.00937
- J. Lucas *et al.*, "Understanding Posterior Collapse in Generative Latent Variable Models", ICLR 2019 proceeding
- A. Razavi *et al.*, "Preventing Posterior Collapse with delta-VAEs", arXiv:1901.03416

4. NORMALIZING FLOWS

A family of methods for constructing flexible learnable probability distributions

Why limit ourselves to just one sample?



from “Why I Stopped Using GAN — ECCV 2020”

The basic idea: change of variables

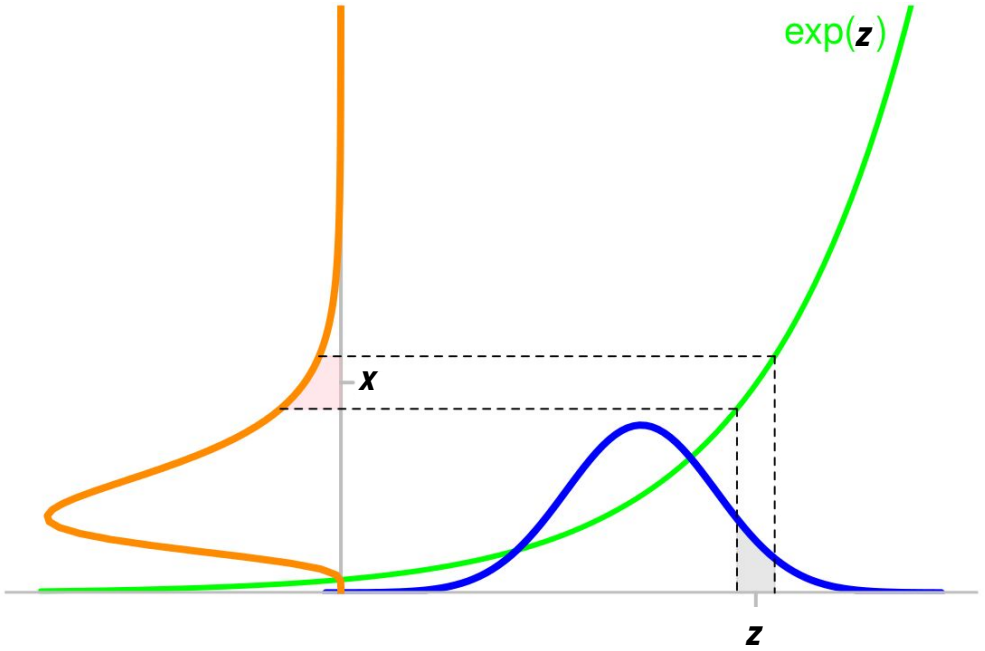
We define a transform *f* such that:

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

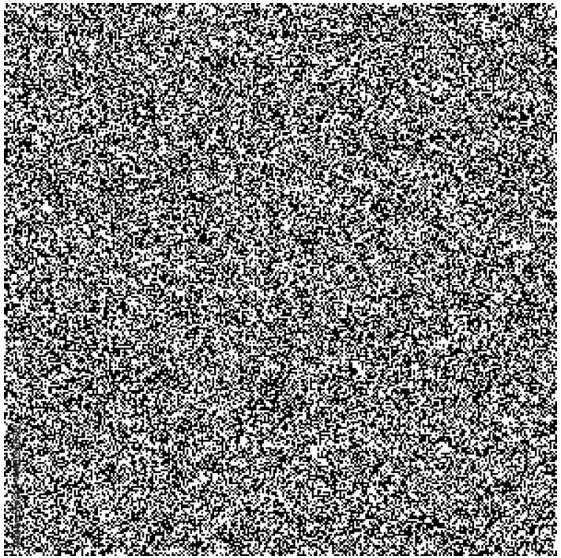
The two pdfs are related:

$$p_x(\mathbf{x})d\mathbf{x} = p_z(\mathbf{z})d\mathbf{z}$$

$$p_x(\mathbf{x}) = p_z(f^{-1}(\mathbf{x}))\det\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right|$$

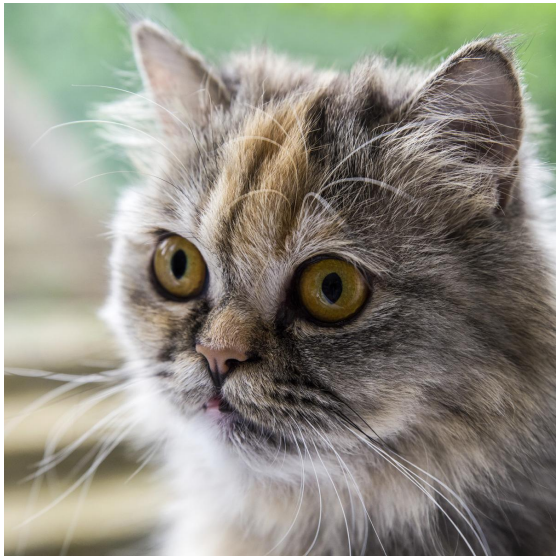


The basic idea: image generation



$f(\mathbf{z})$

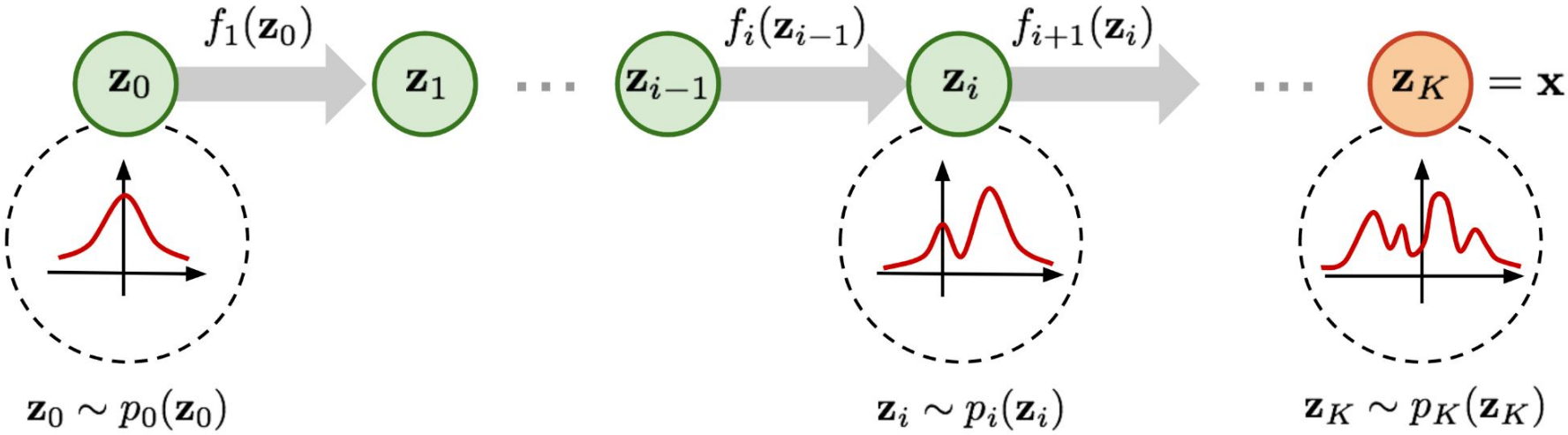
$f^{-1}(\mathbf{x})$



$p_{\mathbf{z}}(\mathbf{z})$

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

The basic idea: complex transforms



from Lilian Weng

We need few building blocks

Task

Learn the ***f(z)*** to send $p_{\mathbf{Z}}\left(\mathbf{Z}\right)$ into the (***unknown***) data distribution $p_{\mathbf{X}}\left(\mathbf{X}\right)$

Pieces

- Basic distribution $p_{\mathbf{Z}}\left(\mathbf{Z}\right)$, typically Gaussian
- Function called flow ***f(z) invertible*** and ***differentiable***, with ***tractable jacobian***

The usage is straightforward

Density evaluation

$$p_x(\mathbf{x}) = p_z(f^{-1}(\mathbf{x})) \det \left| \frac{d\mathbf{z}}{d\mathbf{x}} \right|$$

Sampling new data

- Sample from $p_{\mathbf{z}}(\mathbf{z})$ (Gaussian, trivial)
- Compute $\mathbf{x} = f(\mathbf{z})$ (fast)

The loss function is intuitive

Invertible transform

Jacobian for Volume Correction

$$\log(p_x(x)) = \log(p_z(f^{-1}(\mathbf{x}))) + \log\left(\det \mathbb{J}_{f^{-1}}(\mathbf{x})\right)$$

$$\mathcal{L}(\phi) = -\mathbb{E}_{p_x^*(\mathbf{x})}[\log(p_z(f^{-1}(\mathbf{x};\ \phi))) + \log\left(\det \mathbb{J}_{f^{-1}}(\mathbf{x};\ \phi)\right)]$$

where ϕ are the parameters of *f(z)*

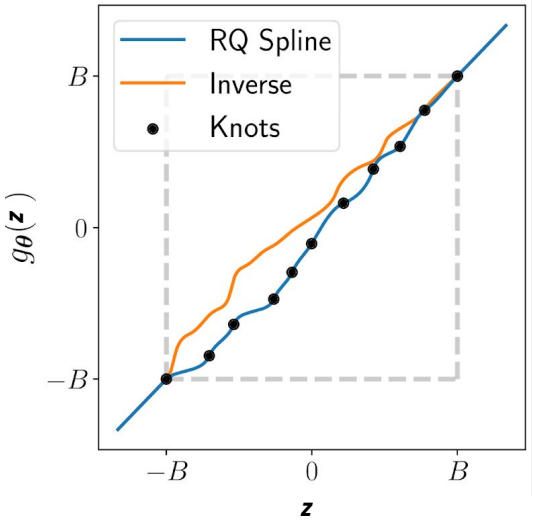
Splines are a smart choice for *f(z)*

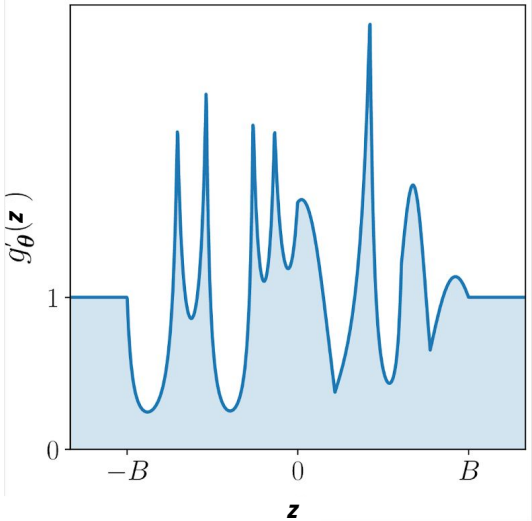
Expressive

Admit analytical inverse,
fast to invert AND evaluate

We use ML to learn the optimal disposition
of points and derivatives

(Just one of the possible choices!)





Normalizing Flows are *powerful* GMs!

Efficient to sample from $p_{\mathbf{x}}(\mathbf{x})$

Efficient to evaluate $p_{\mathbf{x}}(\mathbf{x})$

Highly expressive

Useful latent representation

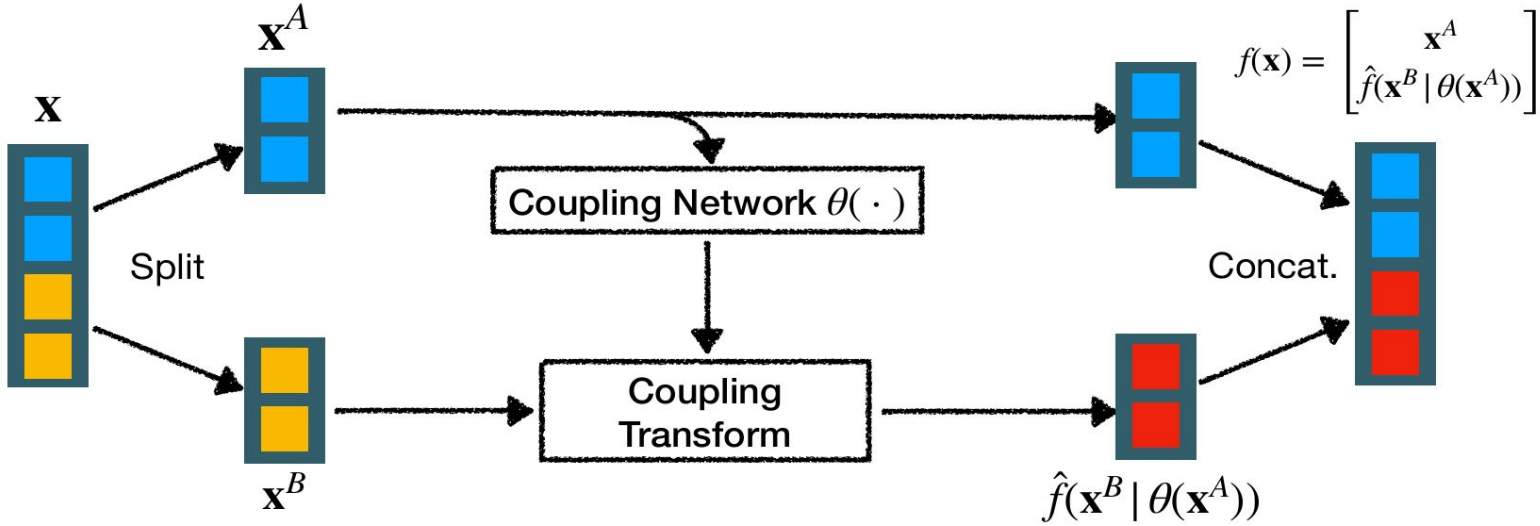
Straightforward to train

Normalizing Flows are *flawed* GMs!

Computation of the Jacobian is hard

Not defined to work in a *discrete* contex!

Coupling layers address Jacobian complexity



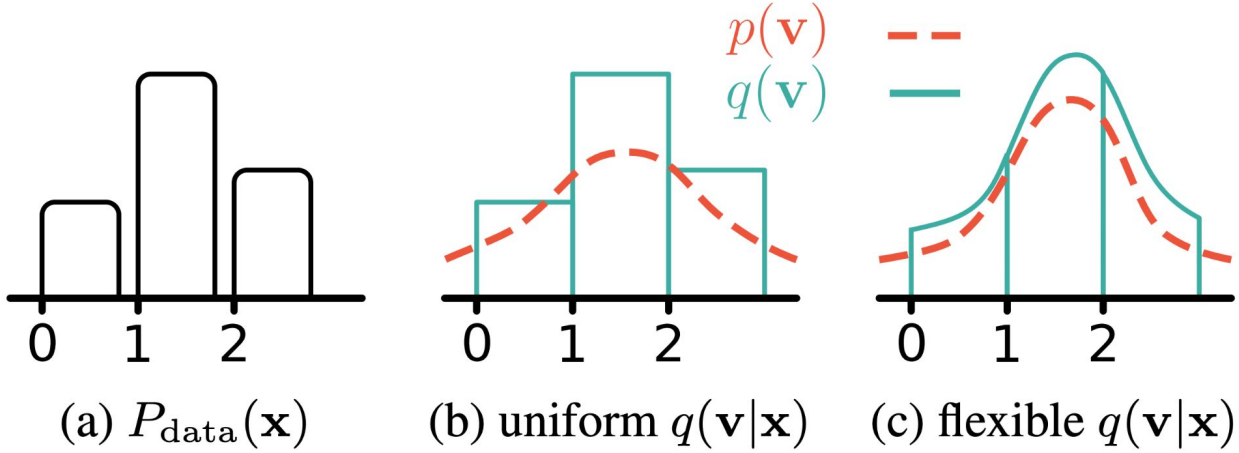
$$\mathbb{J}_f(\mathbf{x};\ \phi) = \begin{pmatrix} \frac{\partial \vec{x}_{1:d}}{\partial \vec{z}_{1:d}} & \frac{\partial \vec{x}_{1:d}}{\partial \vec{z}_{d+1:D}} \\ \frac{\partial \vec{x}_{d+1:D}}{\partial \vec{z}_{1:d}} & \frac{\partial \vec{x}_{d+1:D}}{\partial \vec{z}_{d+1:D}} \end{pmatrix} = \begin{pmatrix} \mathbb{I} & 0 \\ A & \mathbb{J}^* \end{pmatrix}$$

the Jacobian becomes triangular!

from Jason Yu

Dequantization can be used to tackle discrete variables

Apply a gaussian smearing converting discrete data into a continuum



(a) $P_{\text{data}}(\mathbf{x})$ (b) uniform $q(\mathbf{v}|\mathbf{x})$ (c) flexible $q(\mathbf{v}|\mathbf{x})$

from arXiv:2001.11235

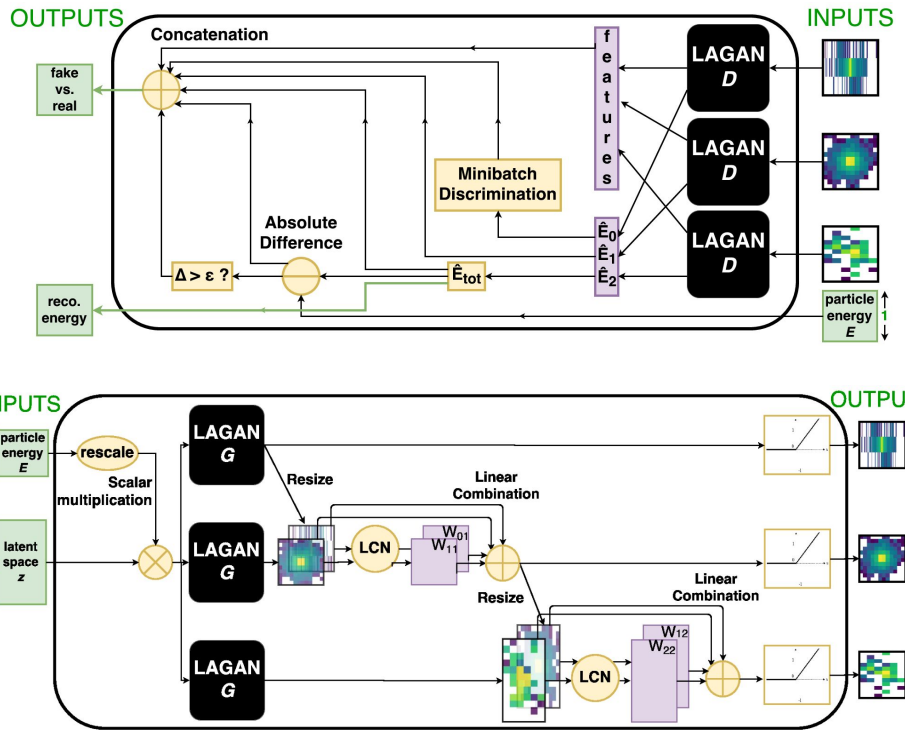
References for Normalizing Flows

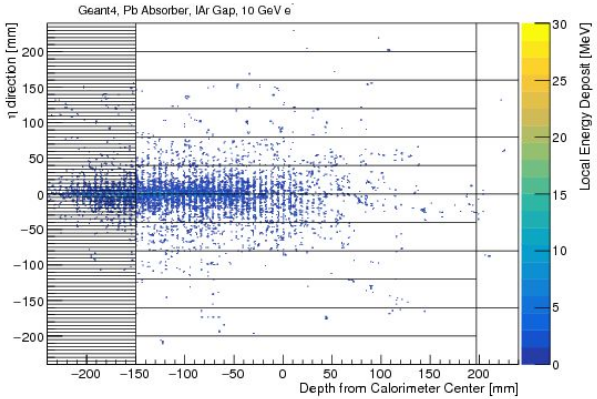
- G. Papamakarios et al, "Normalizing Flows for Probabilistic Modeling and Inference", arXiv:1912.02762
  - An updated and comprehensive review of Normalizing Flows architectures and applications
- C. Durkan et al, "Neural Spline Flows", arXiv:1906.04032
  - The implementation of Normalizing Flows through Rational Quadratic Splines
- Pyro Team, "Normalizing Flows-Introduction", link
  - A cool hands-on introduction to Normalizing Flow in the Pyro package (python). As of this date, there isn't a definitive package for NF neither for Tensorflow nor for Pytorch

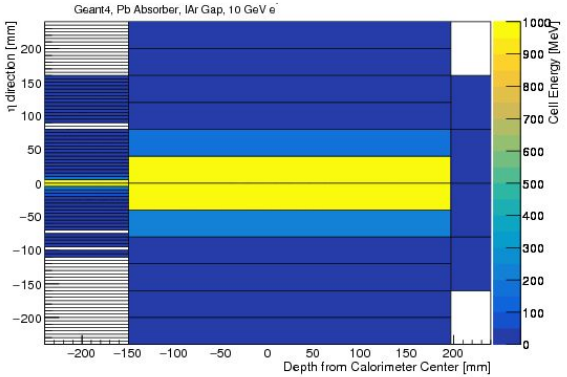
5. PHYSICS USE-CASES

How generative models are used in our field

ML based Fast Simulation: GAN

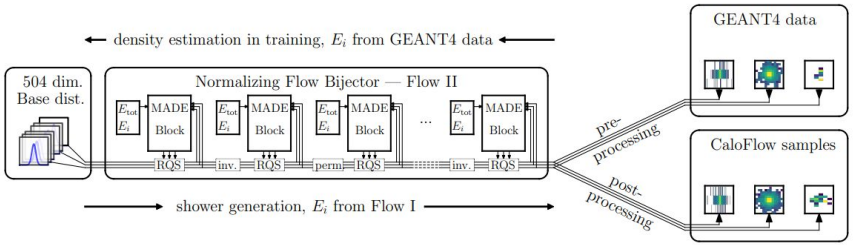


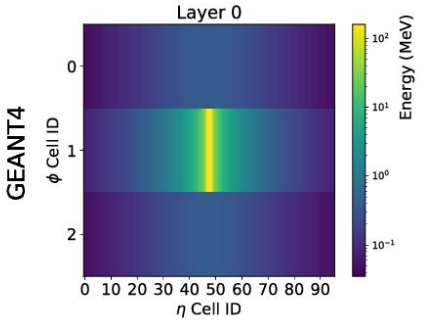


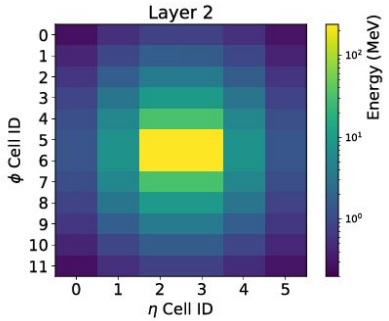


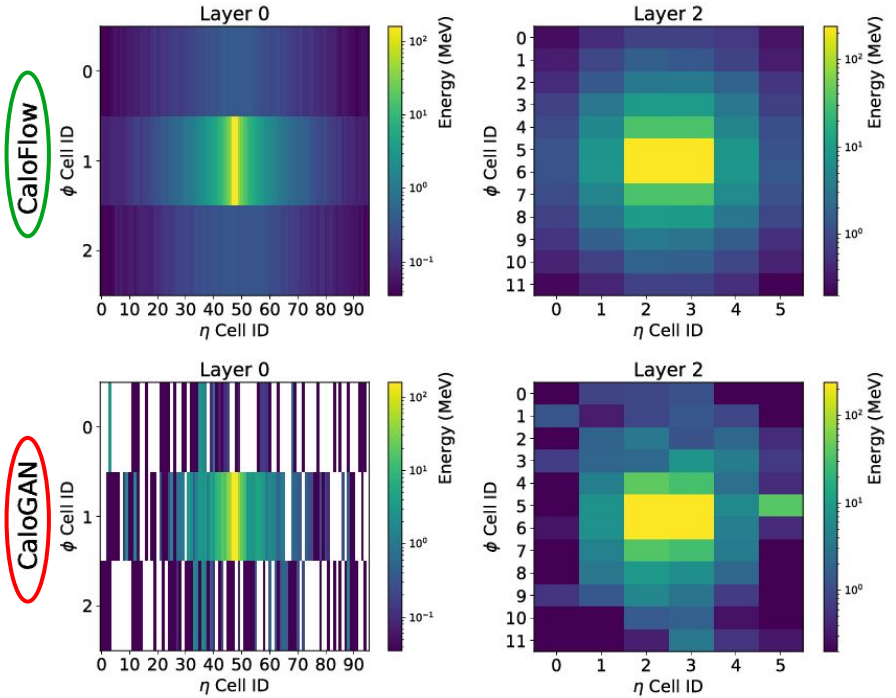
from arXiv:1712.10321

ML based Fast Simulation: Flow





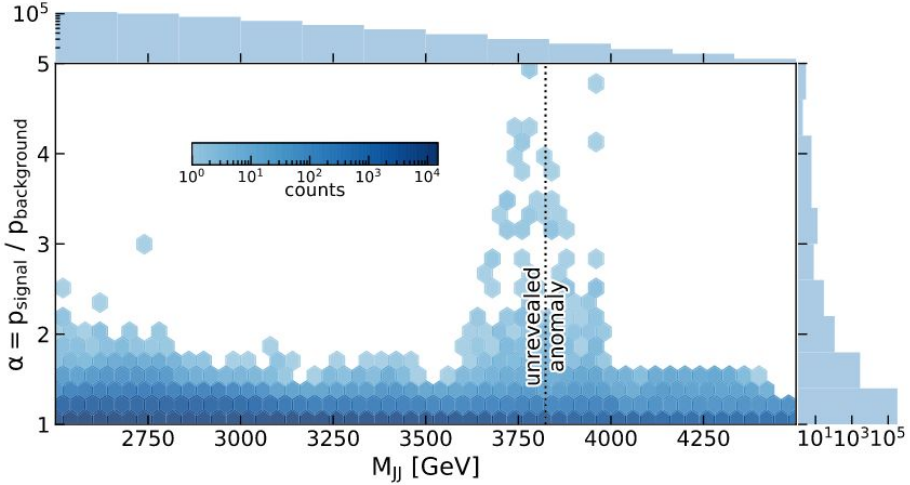


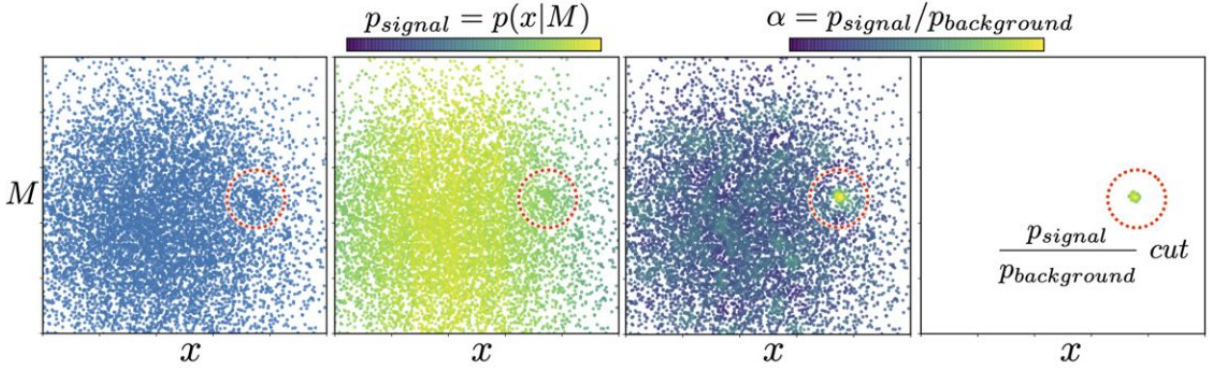


from arXiv:2008.03312

Anomaly Detection

Idea: Model pdf of signal and background using Normalizing Flows





$$p_{bkg} = p(x|M \pm \Delta)$$

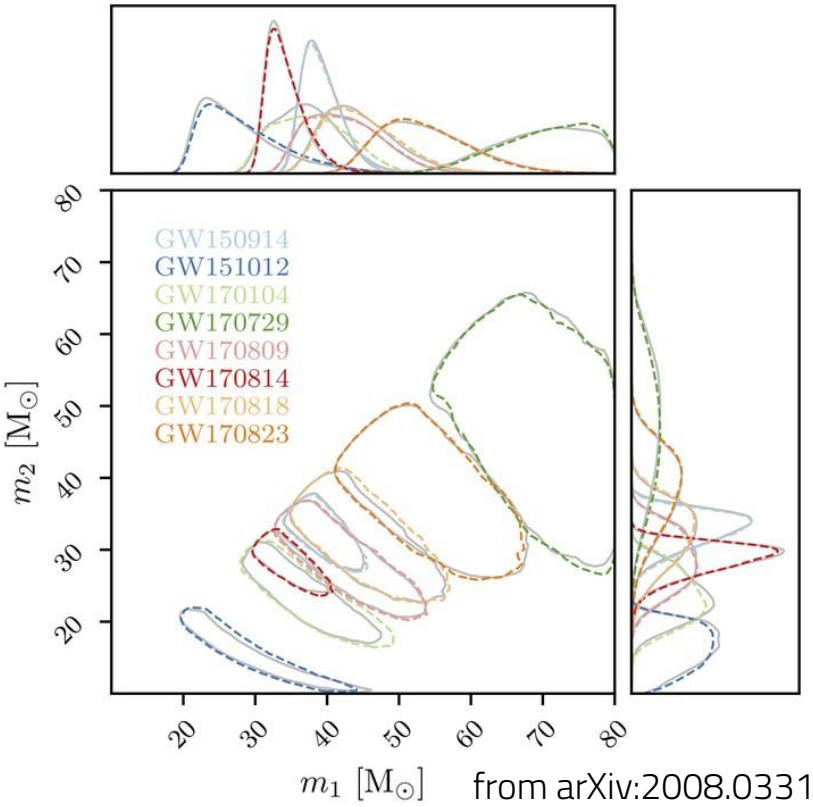
from LHC Olympics 2020

GW Likelihood Free Inference

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)}$$

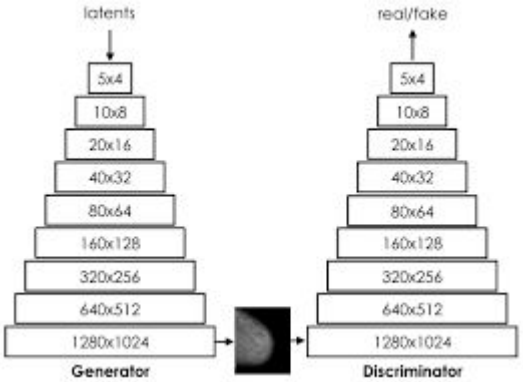
$$(\theta^{(i)}, d^{(i)}) \sim p(\theta, d)$$

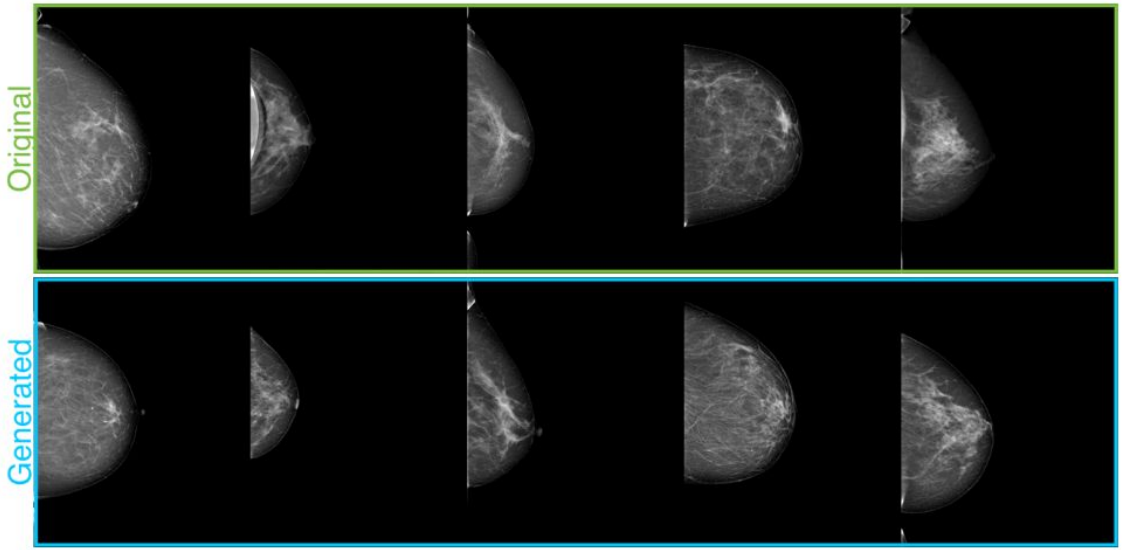
$$q(\theta|d) = \mathcal{N}(0,1)(f_d^{-1}(\theta)) \left|\det J_{f_d^{-1}}\right|$$



from arXiv:2008.03312

ML Generation for Mammograms





from arXiv:1807.03401

References for Physics use-cases

- M. Paganini *et al.*, "CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks", arXiv:1712.10321
- Claudius Krause, David Shih, "CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows", arXiv:2106.05285
- The LHC Olympics 2020: A Community Challenge for Anomaly Detection in High Energy Physics, arXiv:2101.08320
- Stephen R. Green, Jonathan Gair, "Complete parameter inference for GW150914 using deep learning", arXiv:2008.03312
- D. Korkinof *et al.*, "High-Resolution Mammogram Synthesis using Progressive Generative Adversarial Networks", arXiv:1807.03401

CONCLUSIONS

Generative models are a powerful tool at our disposal, with widespread adoption in many Physics use-cases and convincing results

Different models have specific advantages and drawbacks

No readily available implementations for our problems, need to experiment

THANKS!

Any questions?

You can find us at:

- Matteo.Barbetti@fi.infn.it
- Francesco.Vaselli@pi.infn.it

BACKUP

Pedagogical explanation of GANs training

1. Minimax game near convergence: P_{gen} is similar to P_{true} and D is a partially accurate classifier;
2. D is trained to discriminate samples from data, converging to optimality;
3. The G update **is driven by D** that points to region that is more likely to be classified as data;
4. After several steps of training, the networks will reach a point at which both cannot improve because D is unable to differentiate between the two distributions.

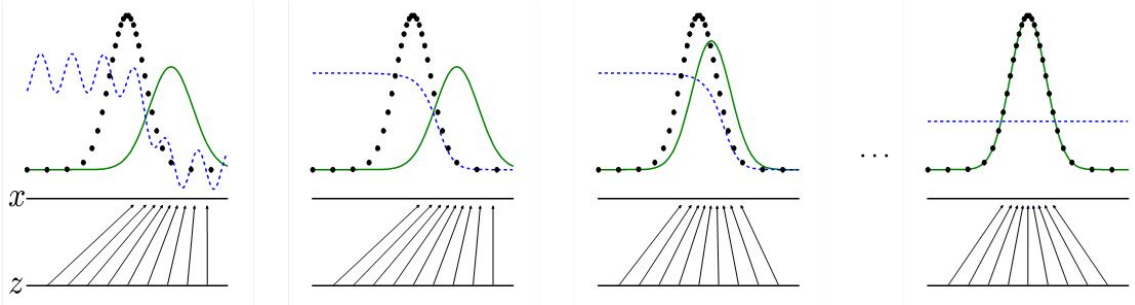


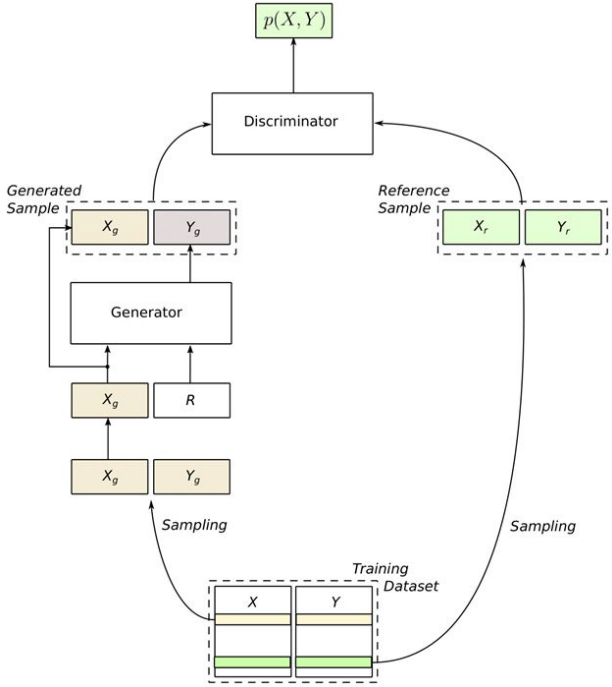
Figure stolen from arXiv:1406.2661

Conditional GANs

GANs provide an easy extension to the **conditional form**. Considering a training sample composed by multi-variable elements, we can imagine to split the variables into:

- variables whose distributions are the goal of the generator (Y)
- variables that simply conditionate the generator output (X)

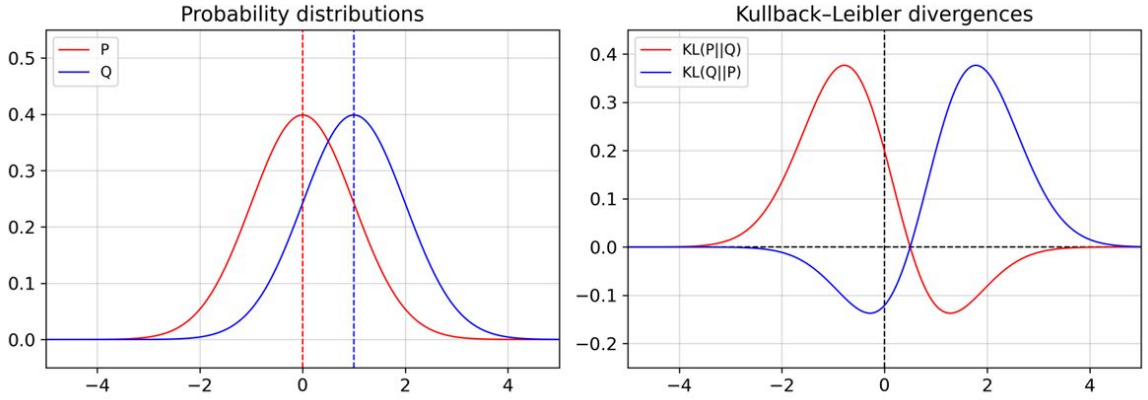
The generator task remains that of reproducing synthetic data (Y), but now the conditional space (X) is joint to the latent space (R) to pursue this goal. On the other hand, the discriminator uses the **generator input/output** to infer the data origin (real or generated).



Kullback-Leibler divergence

The ***Kullback–Leibler divergence*** (also called ***relative entropy***) is a measure of how one probability distribution is different from a second, reference probability distribution.

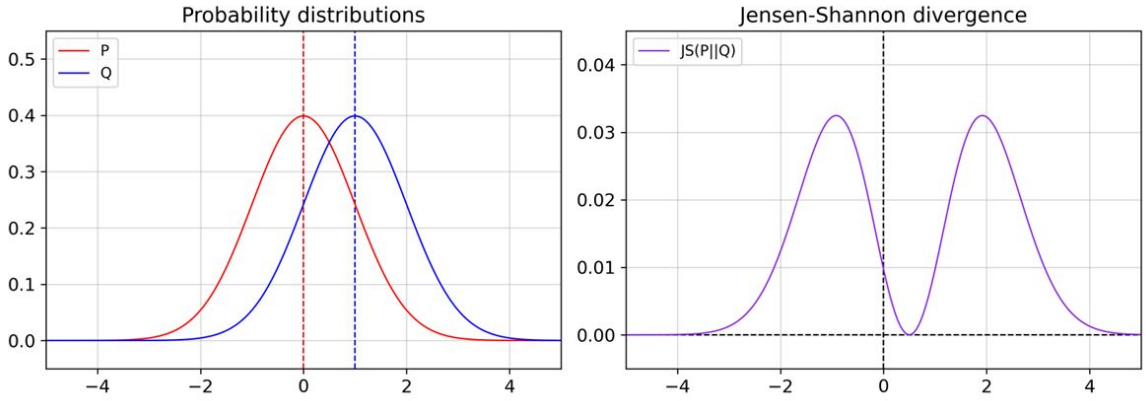
$$D_{KL}(P\|Q) = \mathbb{E}_{x\sim P}[\log P(x)] - \mathbb{E}_{x\sim P}[\log Q(x)]$$



Jensen-Shannon divergence

The ***Jensen–Shannon divergence*** is a method of measuring the similarity between two probability distributions.

$$D_{JS}(P\|Q) = \frac{1}{2}D_{KL}\left(P\|\frac{P+Q}{2}\right) + \frac{1}{2}D_{KL}\left(Q\|\frac{P+Q}{2}\right)$$



Backpropagation in VAEs

In order to ensure that the error correctly *backpropagates* through the nets despite the presence of a random sampling in the halfway of the VAE architecture, a simple trick (called **reparameterization trick**) is used to make the gradient descent possible:

$$z = \mu(x;\phi) + \varepsilon \cdot \sigma(x;\phi) \text{ with } \varepsilon \sim \mathcal{N}(0, I)$$