Generative Models

Elham E Khoda[1] and Aishik Ghosh[2]

[1]University of Washington, [2]University of California, Irvine

Thanks to Sascha Diefenbacher!
part of the presentation is adopted from Sascha's presentation

ML4HEP ICTS 2023
September 4, 2023

UNIVERSITY of WASHINGTON

a3d3 Accelerated AI Algorithms for Data-Driven Discovery

UCI

Generative Models in Media

Learn underlying distribution of data

Produce realistic new samples

Recently gained attention for

- AI art generation (Dall-E 2, Imagen, Midjourney)
- Text Generation (GPT 4, BERT etc)
- AI music generation

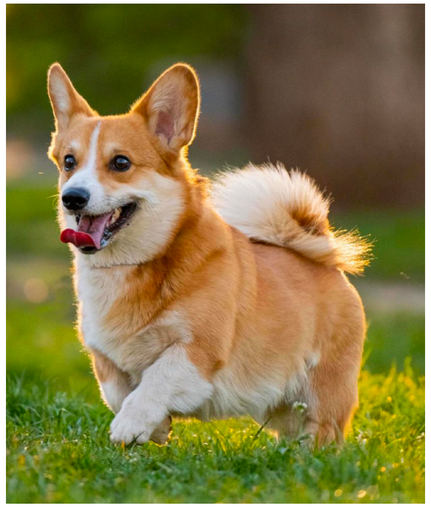
Google Imagen

OpenAI DALL·E 2

particle shower in a calorimeter in the style of Van Gogh, DALL.E 2

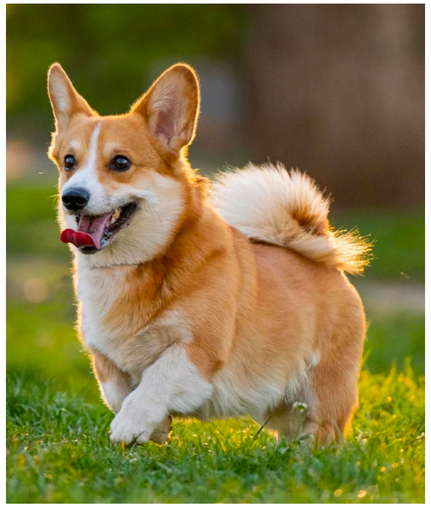
Recap: Classification

High-dimensional data



Recap: Classification

High-dimensional data → **Low-dimensional label**

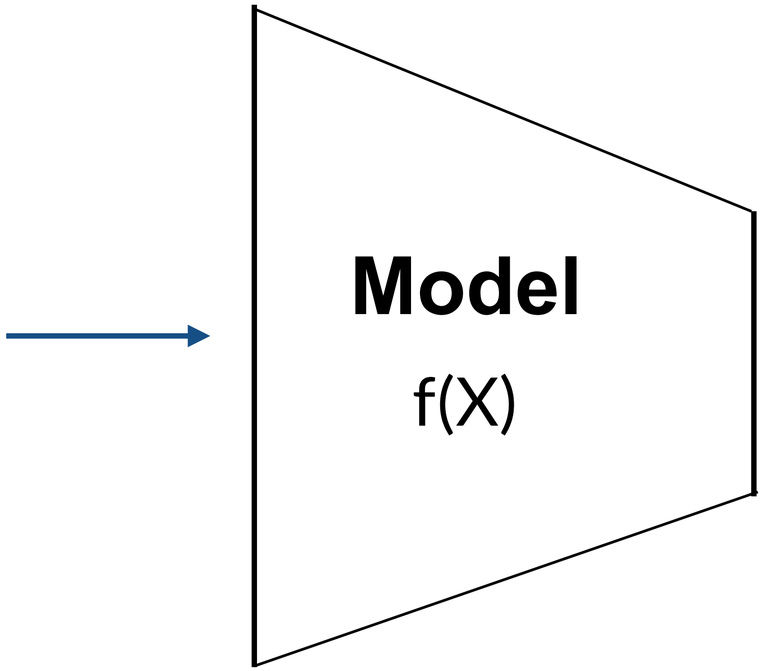


→ Dog

→ Cat

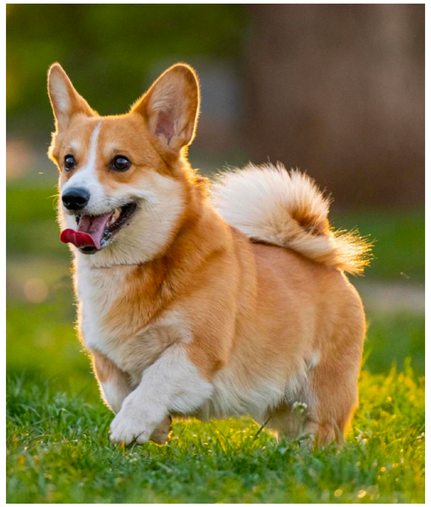
Recap: Classification

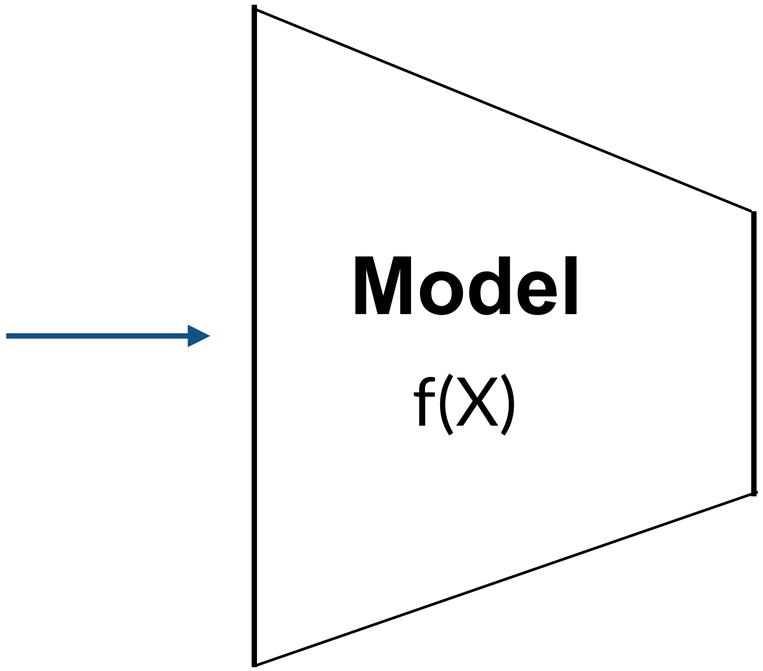
High-dimensional data



Recap: Classification

High-dimensional data





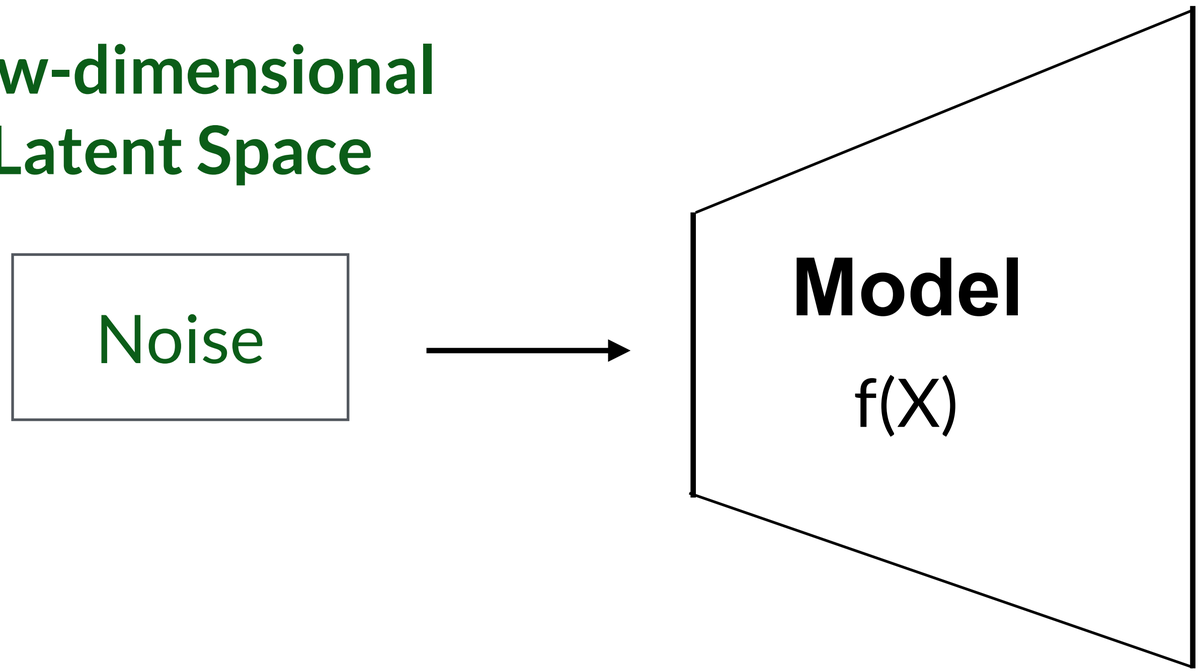
Prediction

Cat / Dog

Generation

We flip this around

Low-dimensional Latent Space

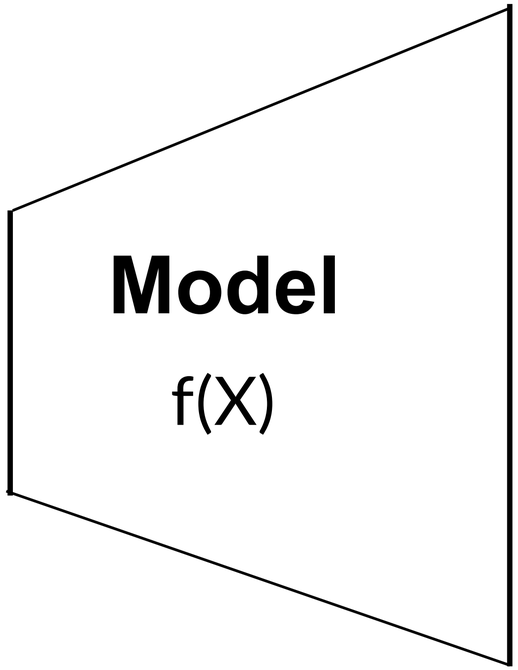


Generation

We flip this around

Low-dimensional Latent Space

Noise

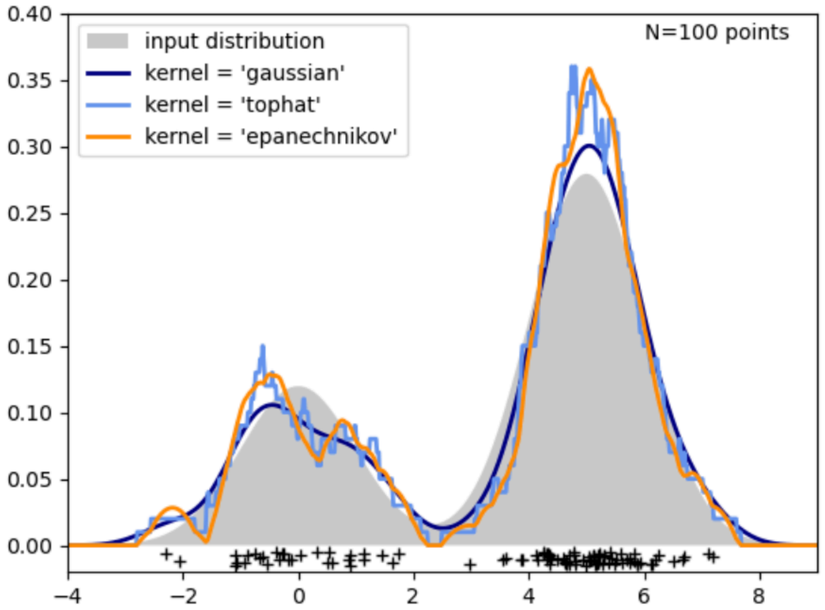


High-dimensional data

Generative Modelling

Take input training samples from some distribution and learn a model that represents the underlying distribution

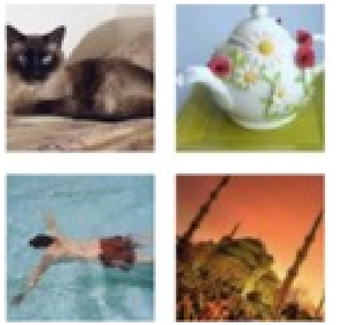
Density Estimation



Sample Generation

Training data

$\sim p_{\text{data}}(x)$



Generated data

$\sim p_{\text{model}}(x)$

How can we learn $\sim p_{\text{model}}(x)$ similar to $\sim p_{\text{data}}(x)$?

Training a generative model

1. Start from random noise

2. Use network to transform noise to data

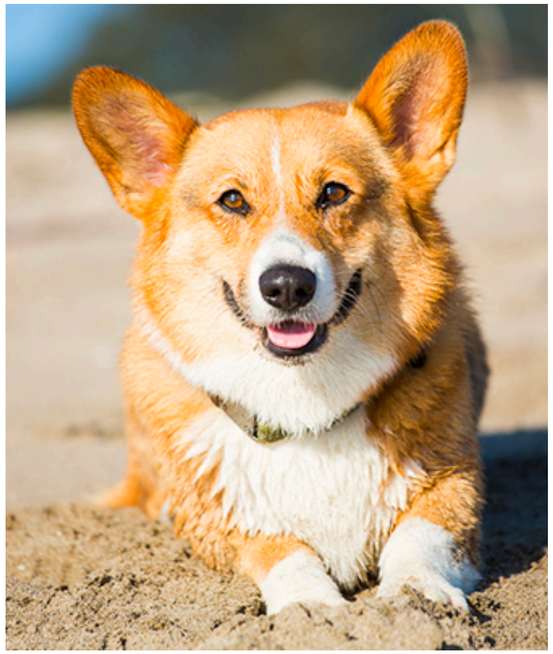
3. Use loss function to monitor the quality of generated data

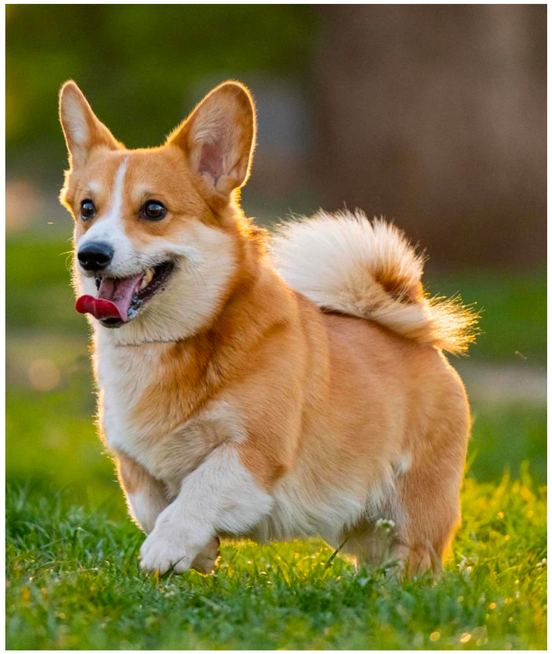
Slide borrowed from Sascha

Generative Loss Function

How do you measure the model performance?

How do you express it mathematically (and differentiable)?





Slide borrowed from Sascha

Difficulties in Generation

Lets assume we have a set of images

Does the new set have the same properties as the training data?

Training Data

Generated Data

Slide borrowed from Sascha

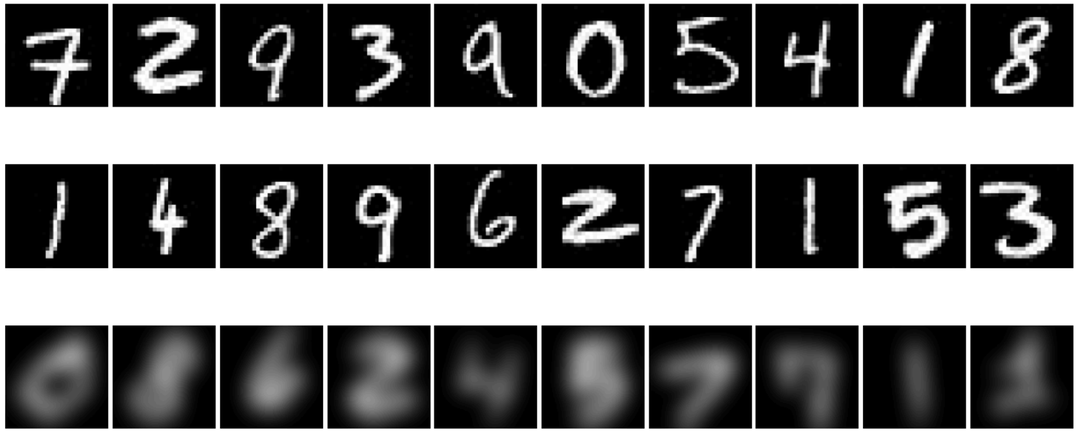
Difficulties in Generation

Input Set

Ideal Generative Output

Slide borrowed from Sascha

Difficulties in Generation



Input Set

Ideal Generative Output

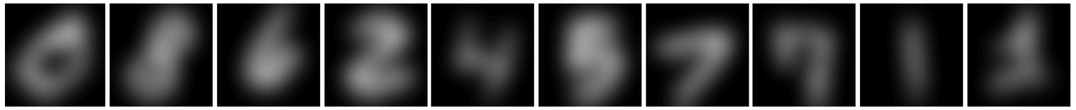
Low Sample Quality

Slide borrowed from Sascha

Difficulties in Generation

Input Set

Ideal Generative Output

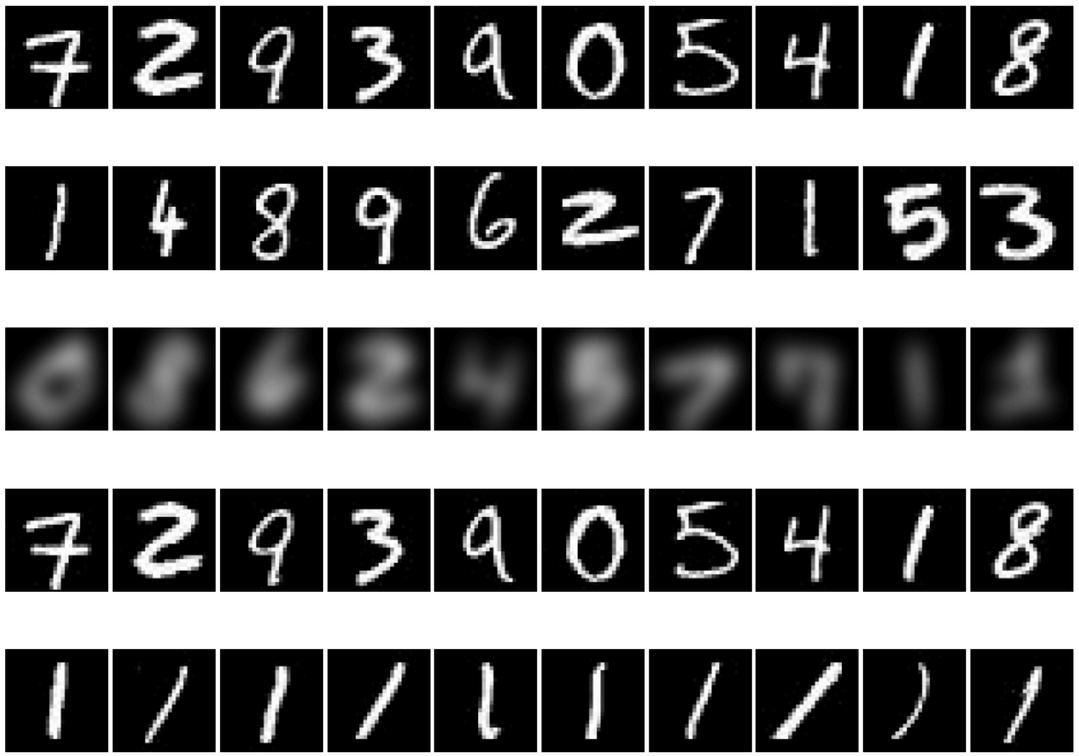


Low Sample Quality

Overfitting

Slide borrowed from Sascha

Difficulties in Generation



Input Set

Ideal Generative Output

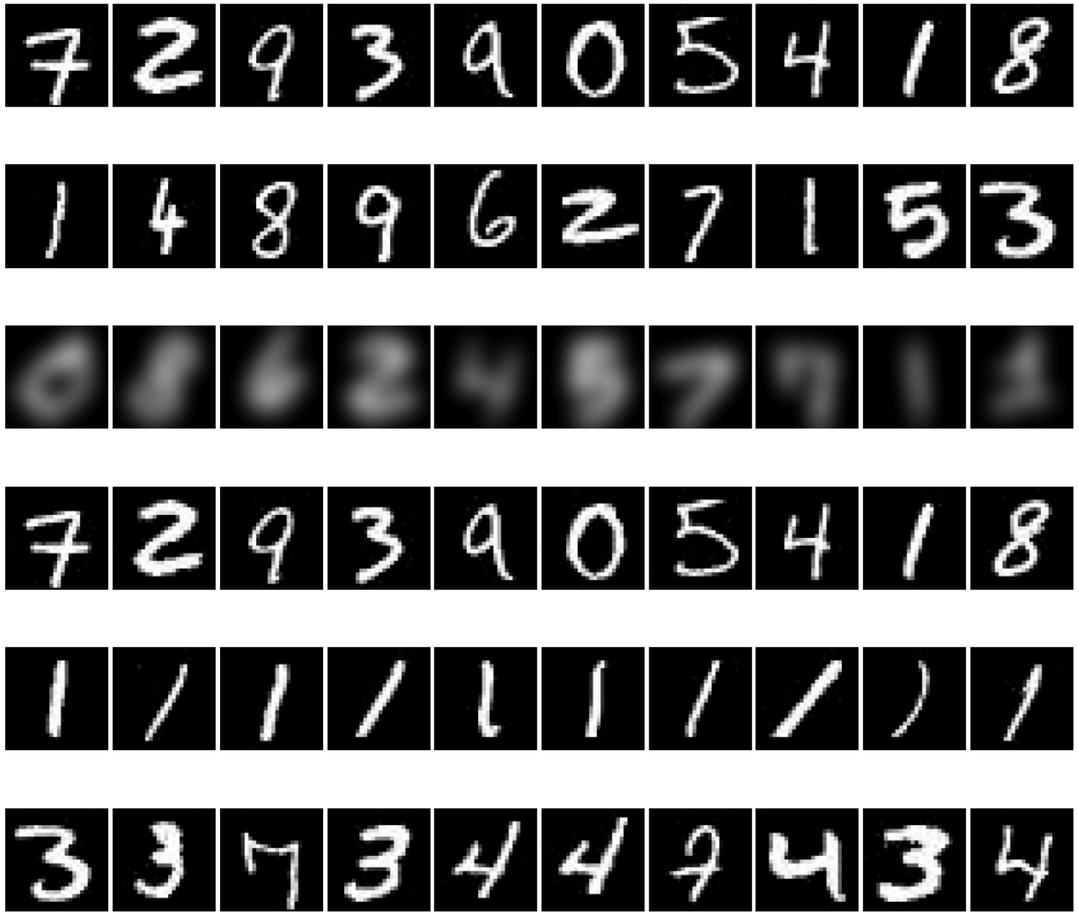
Low Sample Quality

Overfitting

Mode Collapse

Slide borrowed from Sascha

Difficulties in Generation



Input Set

Ideal Generative Output

Low Sample Quality

Overfitting

Mode Collapse

Incorrect Composition

Generative Models

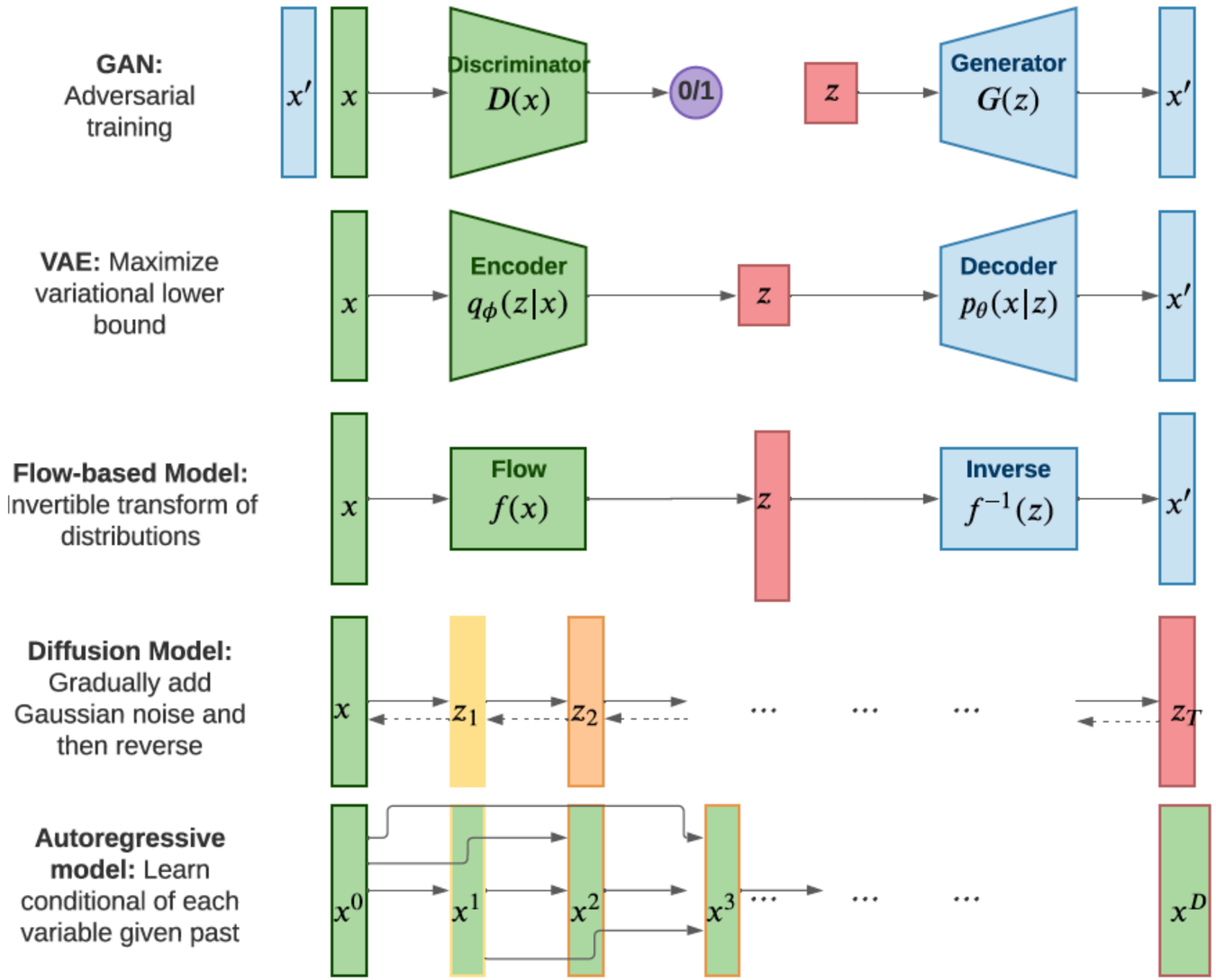


Figure credit: Lilian Weng

KL (Kullback–Leibler) divergence

$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

it is the expectation of the logarithmic difference between the probabilities p and q, where the expectation is taken using the probabilities p.

What is the minima of KL divergence ?
At what condition that happens?

Jensen–Shannon (JS) Divergence

JS divergence is another measure of similarity between two probability distributions, bounded by [0, 1]

$$D_{JS}(p\|q) = \frac{1}{2}D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q\|\frac{p+q}{2})$$

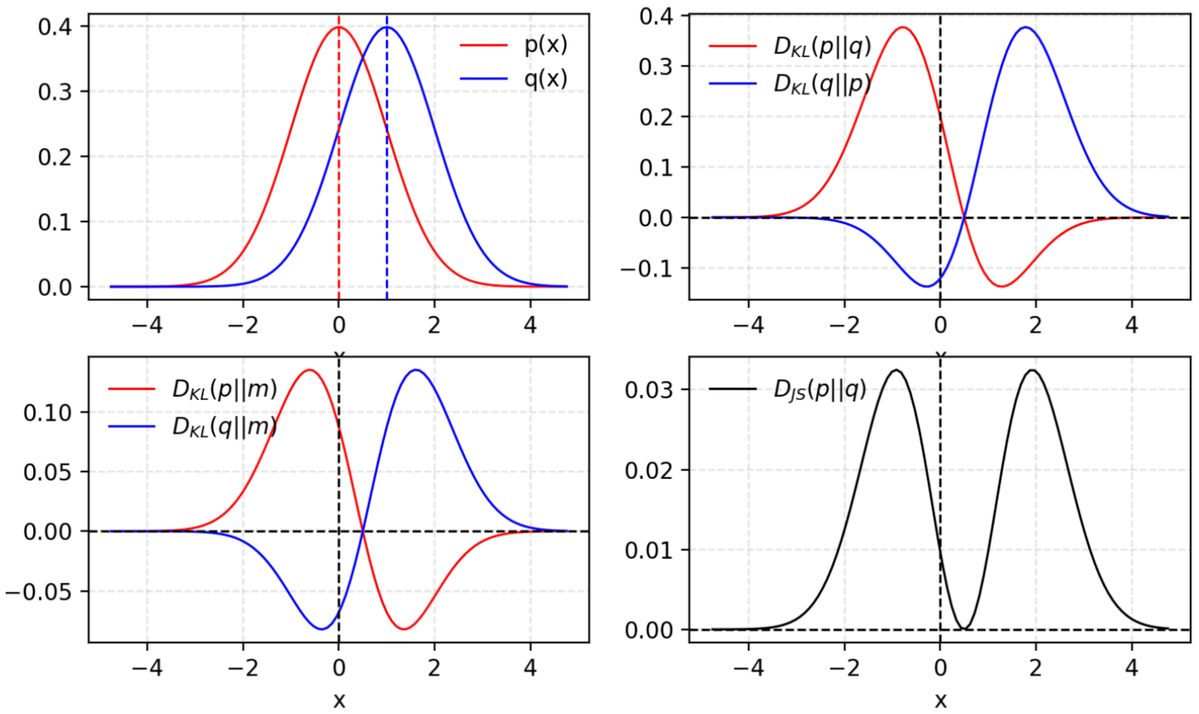


Fig. 1. Given two Gaussian distribution, p with mean=0 and std=1 and q with mean=1 and std=1. The average of two distributions is labelled as $m = (p + q)/2$. KL divergence D_{KL} is asymmetric but JS divergence D_{JS} is symmetric.

Source

Generative Adversarial Network (GAN)

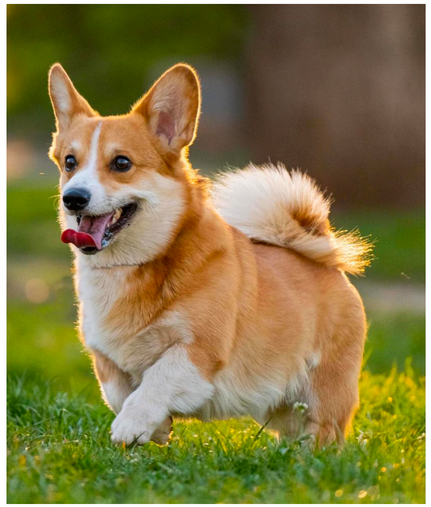
Low-dimensional Latent Space

Noise

Model

f(X)

High-dimensional data



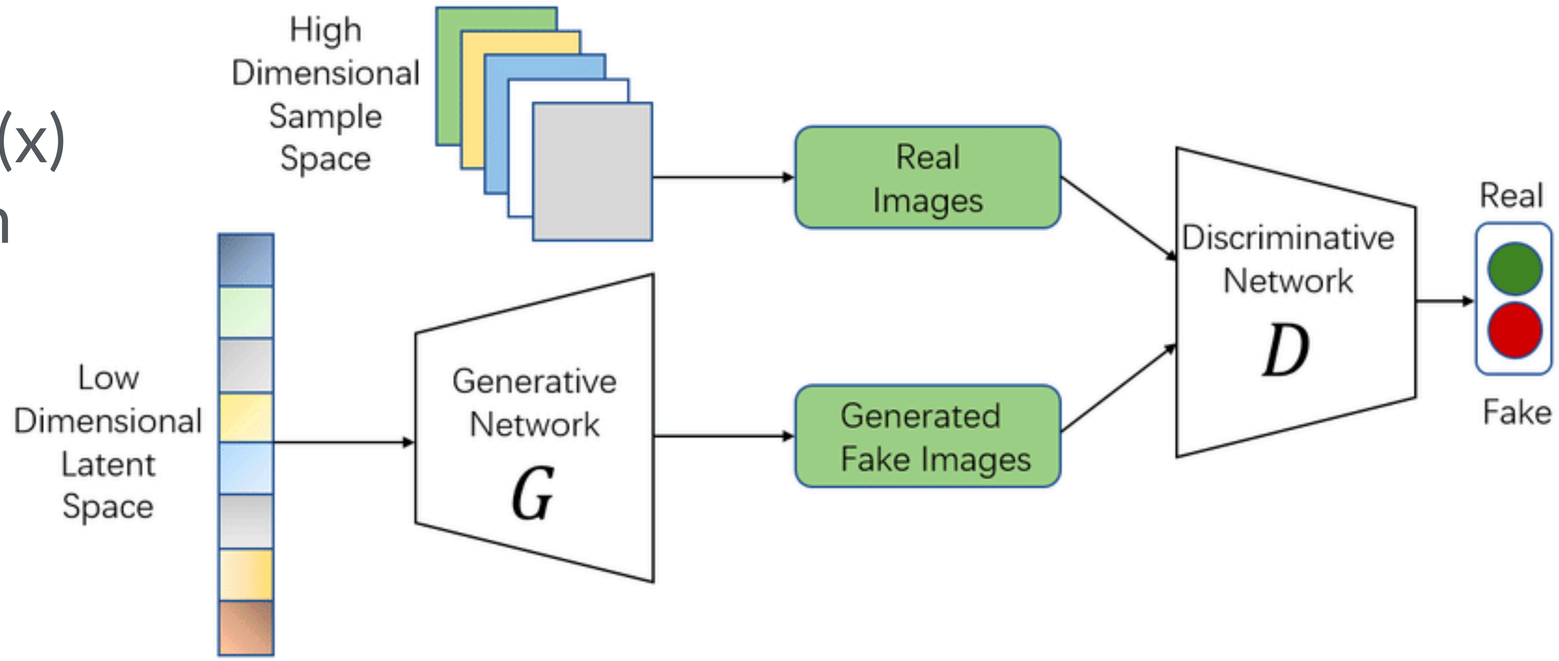
Generative Adversarial Network (GAN)

Generator Network G(z) = x

- Maps noise $z \rightarrow$ Data x

Discriminator D(G(z)) and D(x)

- Learns difference between real and fake



D(G(z)) is differentiable function measuring performance
Use D(G(z)) as loss to update G

GAN: Pros and Cons

Pros

- Intuitive approach
- Easy to introduce additional constraints
- Well explored with several improvements (WGANs, normalisations)

Cons

- Difficult to train
- Generator and Discriminator needs to be balances
- Can fail to converge
- Prone to mode collapse

Variational AutoEncoder

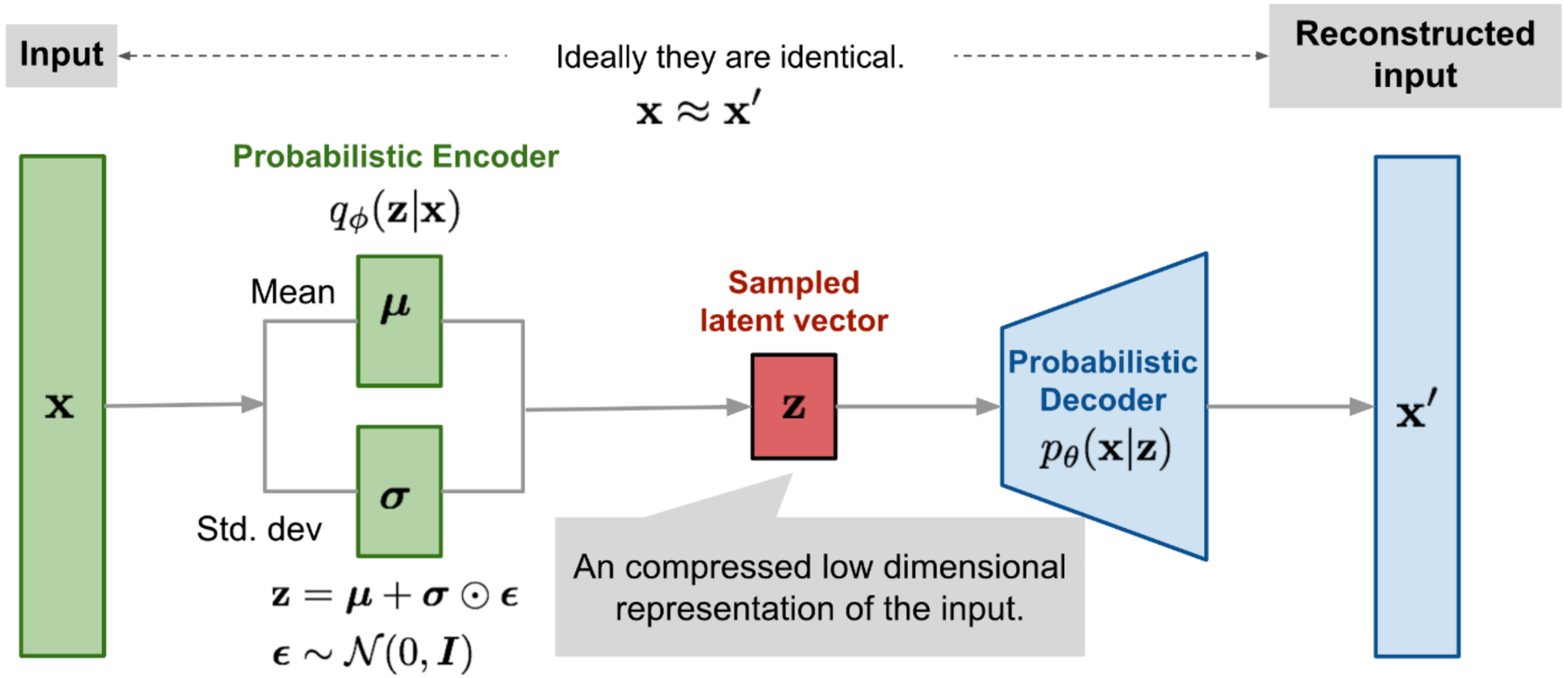


Figure credit: Lilian Weng

AutoEncoder

- ***Encoder* network** g_ϕ : Maps original high-dimension input (x) into the latent low-dimensional latent space, z.

  $g_\phi(x) = z$

- ***Decoder* network** f_θ: Recovers the data from the latent space.

  $f_\theta(z) = x$

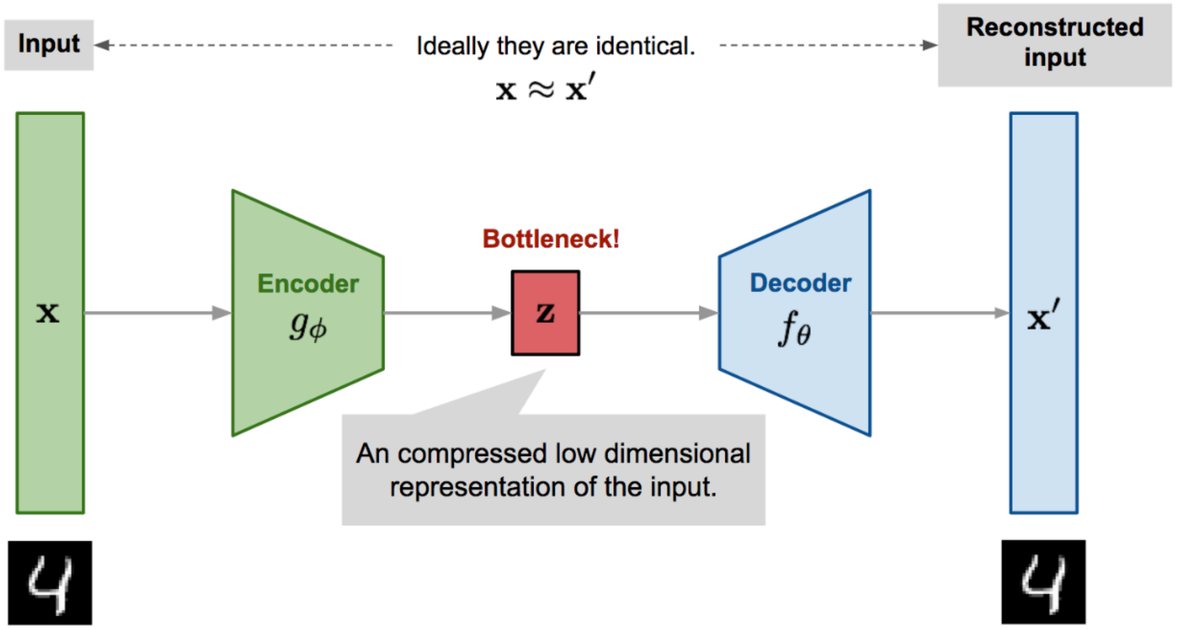
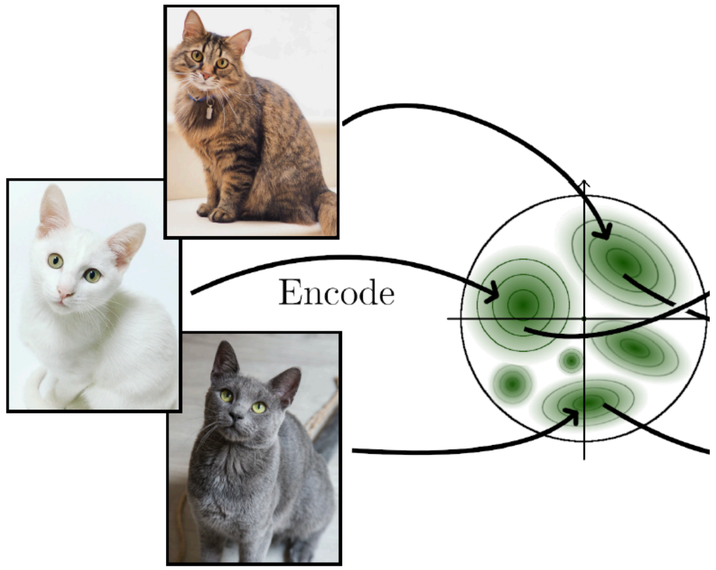


Figure credit: Lilian Weng

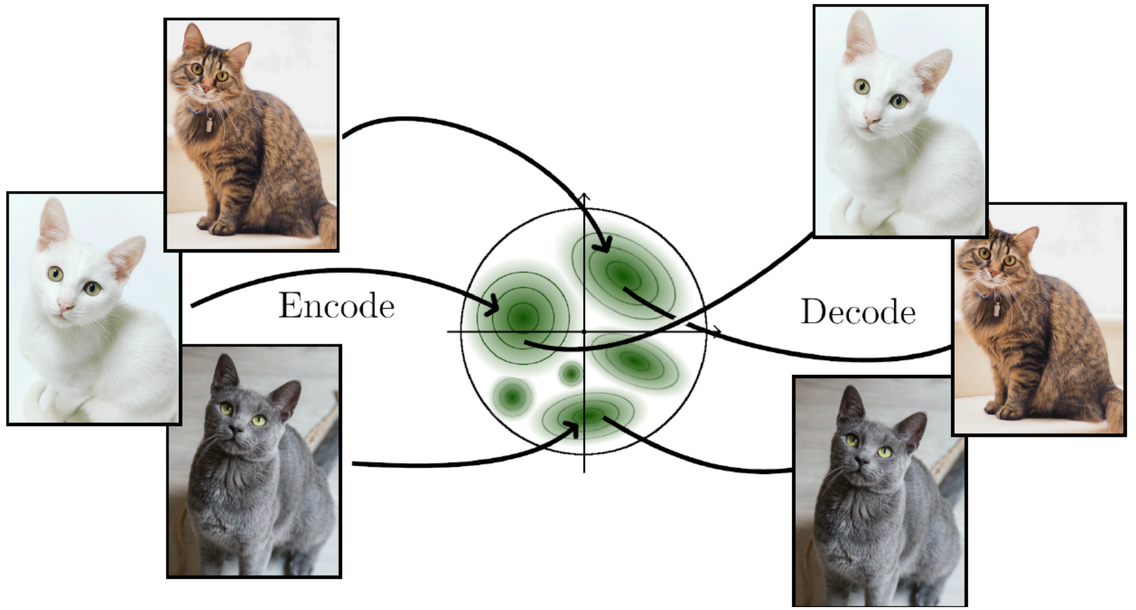
Slide borrowed from Sascha

AutoEncoder



Encoder: Mapping $x \rightarrow z$, $g_\phi(x) = z$

AutoEncoder

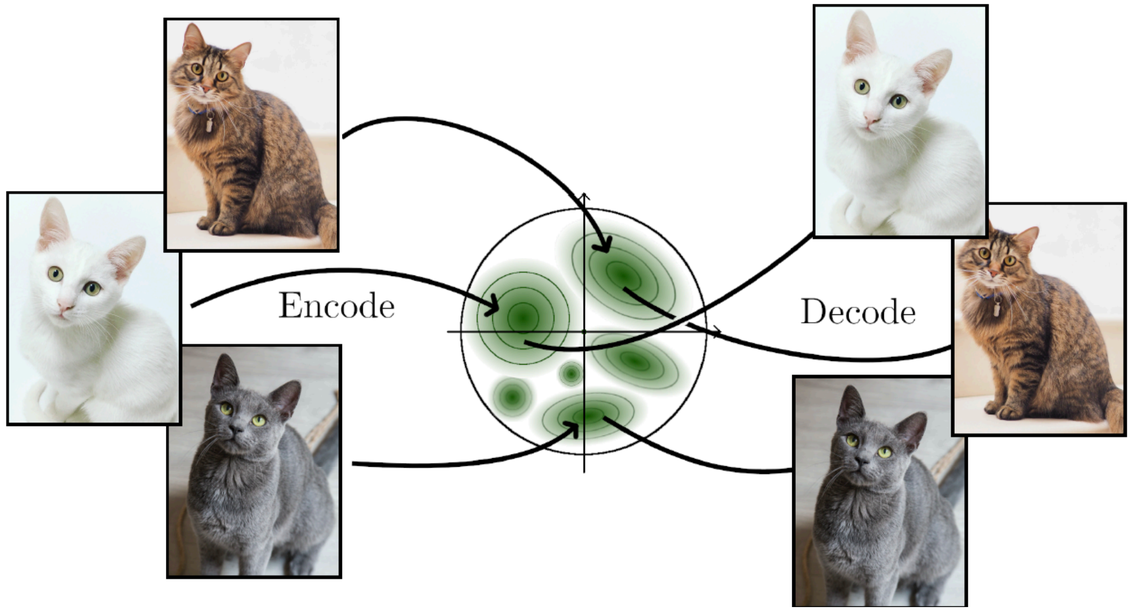


Encoder: Mapping $x \to z$, $g_\phi(x) = z$

Decoder: Mapping $z \to x$, $f_\theta(z) = x$

Slide borrowed from Sascha

AutoEncoder

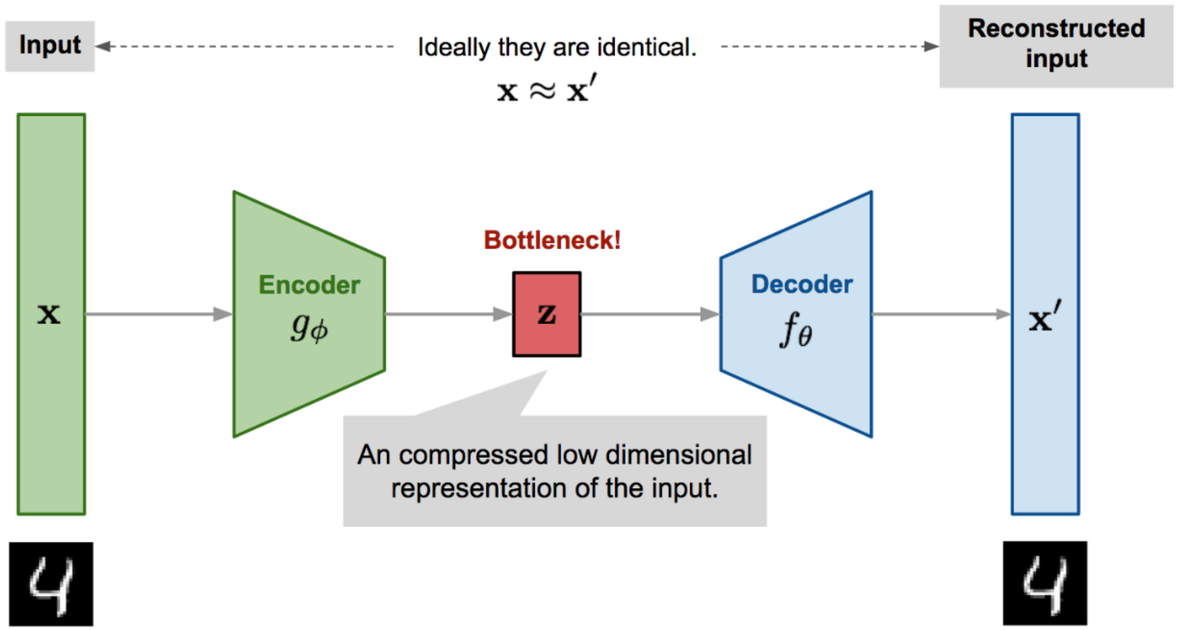


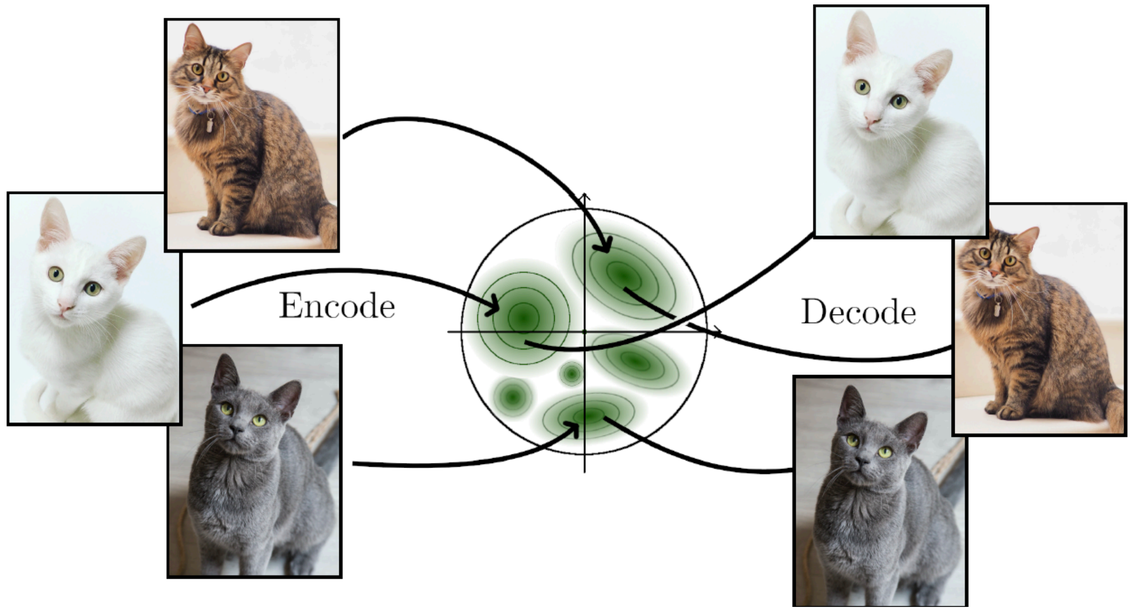
Encoder: Mapping $x \to z$, $g_\phi(x) = z$

Decoder: Mapping $z \to x$, $f_\theta(z) = x$

Compare Input and Output pixel by pixel with mean squared error

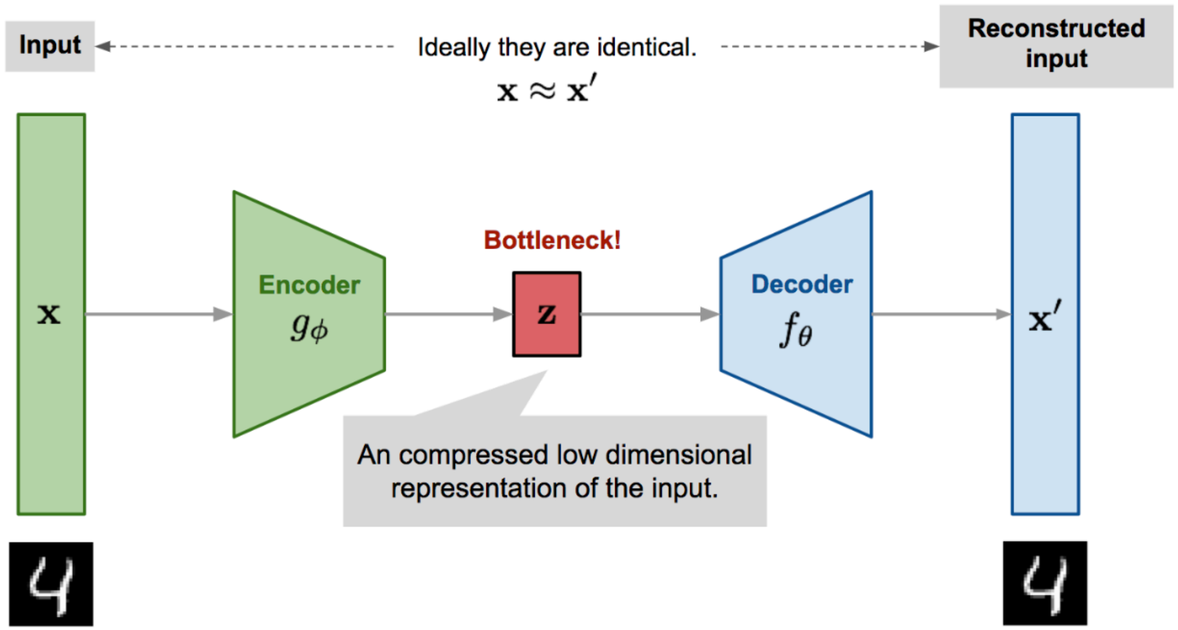
Variational AutoEncoder

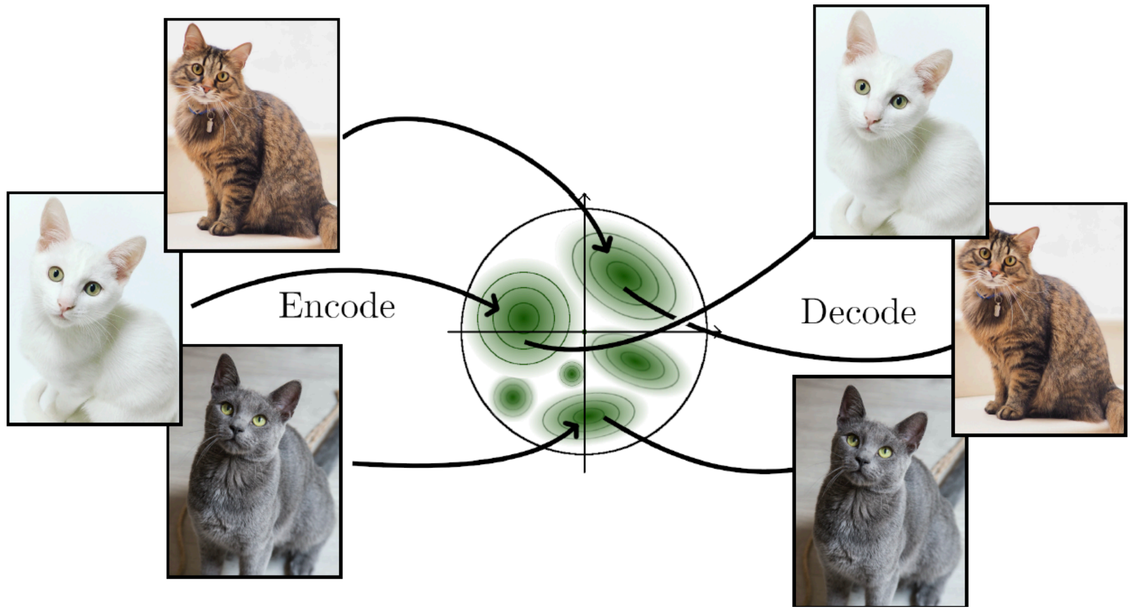




Sample from z and feed it to Decoder $f_\theta(z)$ ⟶ **Generate New Samples?**

Variational AutoEncoder



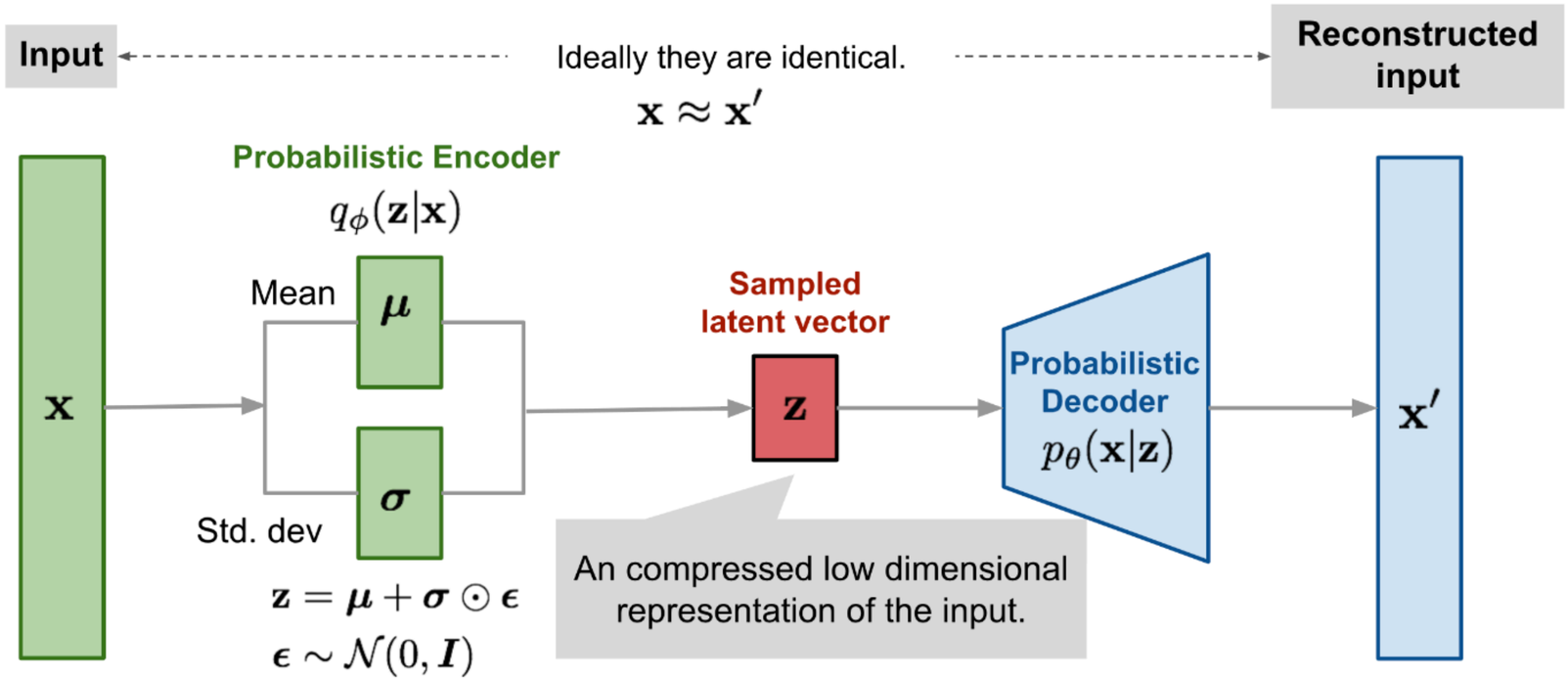


Sample from z and feed it to Decoder $f_\theta(z)$ ⟶ **Generate New Samples?**

Need regularized latent space to sample from

⟶ ***Variational AutoEncoder***

Variational AutoEncoder



Latent space: Series of Gaussians, regularized match to *standard normal N(0,1)*

We can use KL divergence $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) = \sum_i \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$

Compare input and output again using MSE

Figure credit: Lilian Weng

Variational AutoEncoder

Pros

- Directly evaluates log likelihood
- Stable in training

Cons

- MSE loss insufficient for certain data sets
- Needs to balance KLD and MSE loss terms

Jacobian Matrix

Given a function of mapping a n-dimensional input vector $\mathbf{x}$ to a m-dimensional output vector, $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, the matrix of all first-order partial derivatives of this function is called the **Jacobian matrix**, $\mathbf{J}$ where one entry on the i-th row and j-th column is $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Change of Variable Theorem

Given a random variable z and its known probability density function $z \sim \pi(z)$, we would like to construct a new random variable using a 1-1 mapping function $x = f(z)$. The function f is invertible, so $z = f^{-1}(x)$. Now the question is *how to infer the unknown probability density function of the new variable*, $p(x)$?

$$\int p(x)dx = \int \pi(z)dz = 1 \text{ ; Definition of probability distribution.}$$

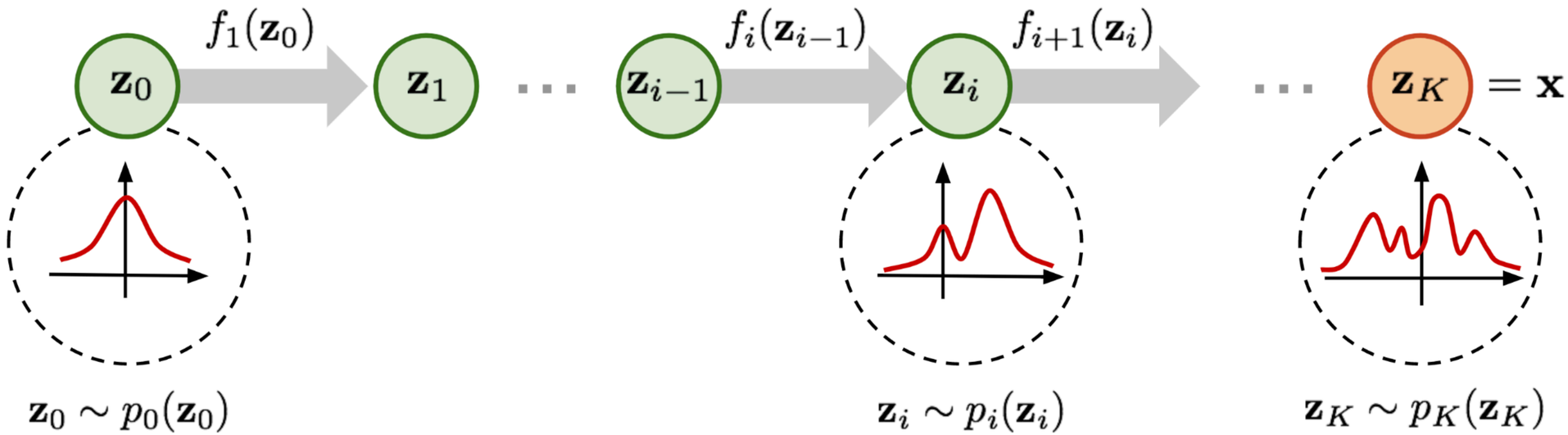
Change of Variable Theorem

Given a random variable z and its known probability density function $z \sim \pi(z)$, we would like to construct a new random variable using a 1-1 mapping function $x = f(z)$. The function f is invertible, so $z = f^{-1}(x)$. Now the question is *how to infer the unknown probability density function of the new variable*, $p(x)$?

$$\int p(x)dx = \int \pi(z)dz = 1 \text{ ; Definition of probability distribution.}$$

$$p(x) = \pi(z)\left|\frac{dz}{dx}\right| = \pi(f^{-1}(x))\left|\frac{df^{-1}}{dx}\right| = \pi(f^{-1}(x))|(f^{-1})'(x)|$$

Normalizing Flows



$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1})$$

$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{ thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|$$

Normalizing Flows: Loss

Loss Function: the negative log-likelihood (NLL) over the training dataset, *D*

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log p(\mathbf{x})$$

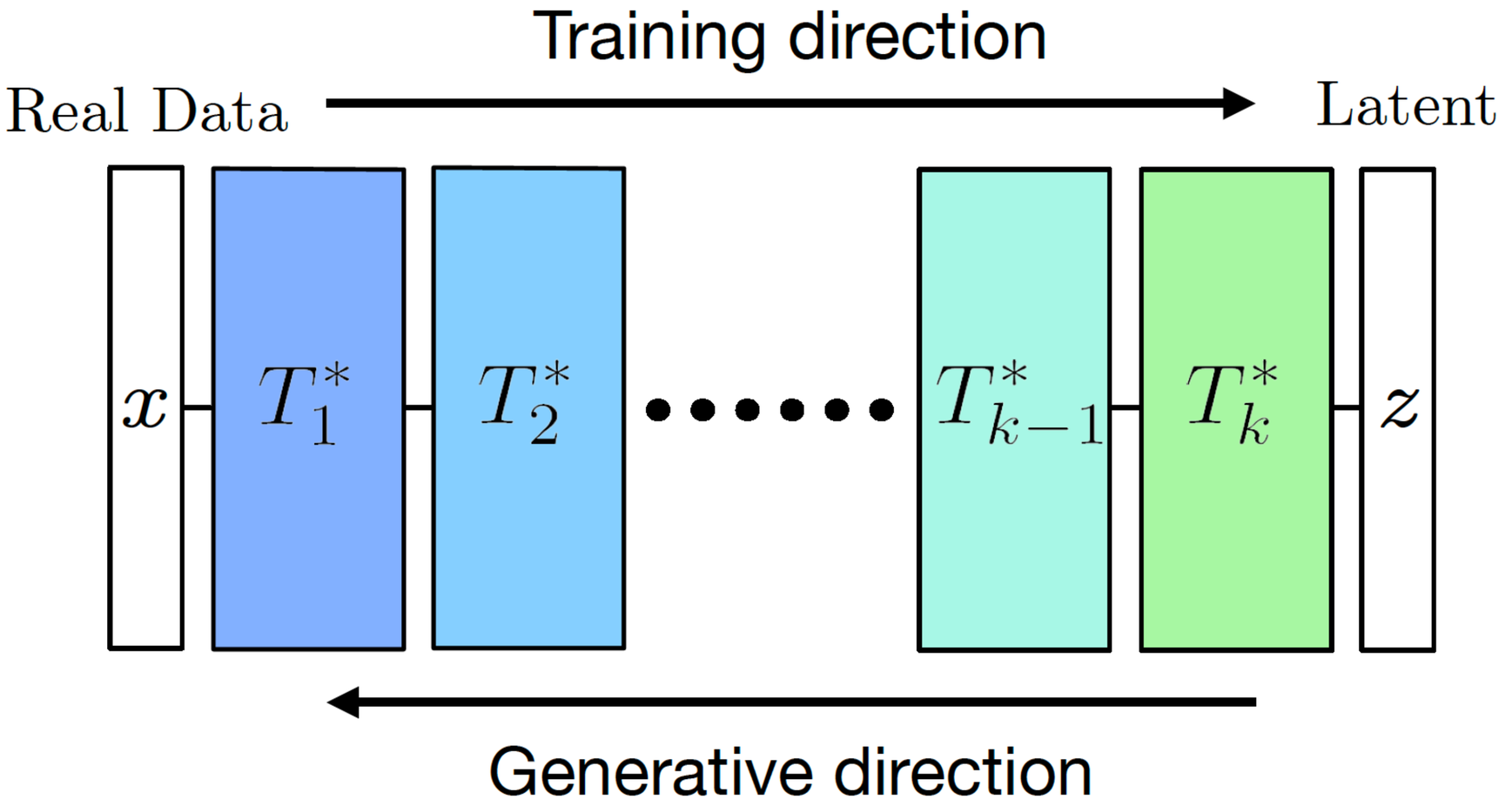
$$\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) \quad = \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|$$

Latent
Normal Distribution

Jacobian
of the transformation

Slide borrowed from Sascha

Normalizing Flows



Normalizing Flows

Pros

- Directly evaluates log likelihood
- Stable in training
- High generative quality
- Easy to train and use

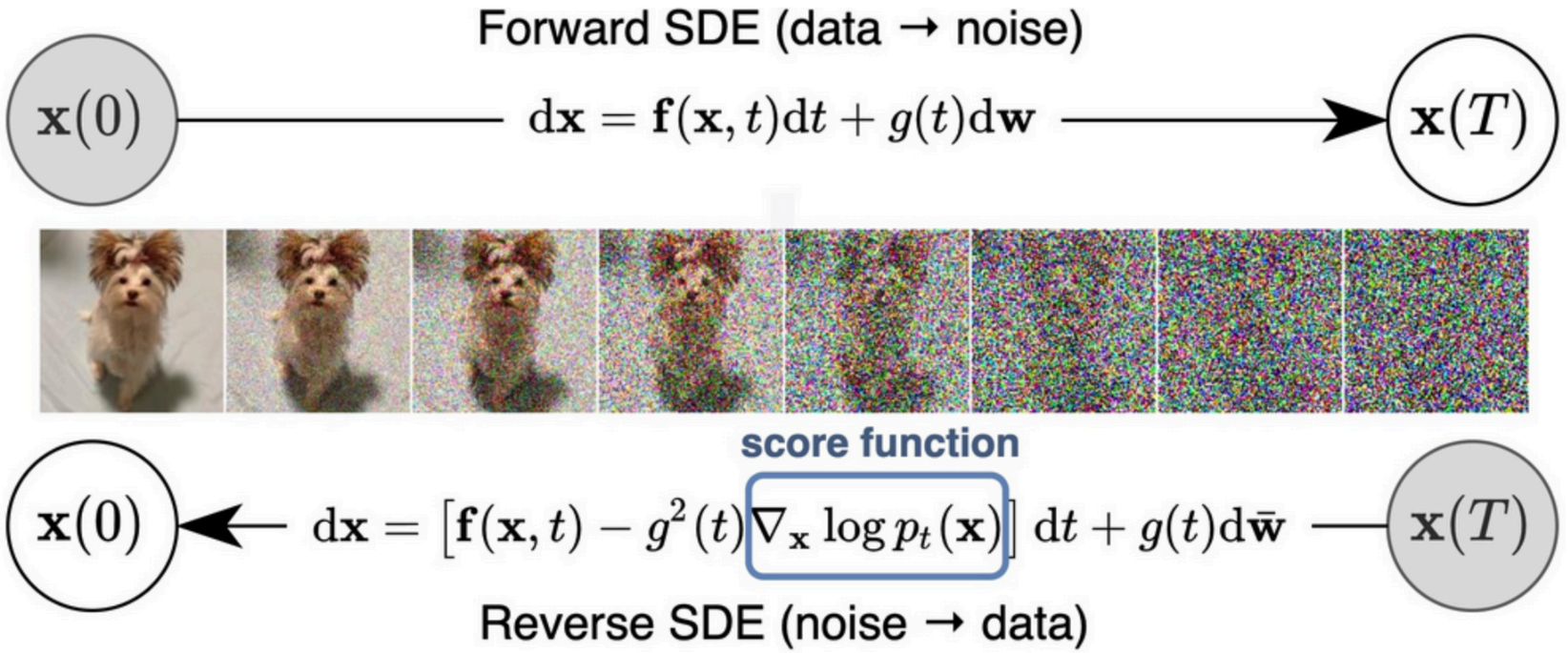
Cons

- Fixed dimensionality through entire flow
- Slow generation times for large models/data

Diffusion Model

Diffusion models are inspired by non-equilibrium thermodynamics

- Repeatedly add noise to data point
- Learn to undo noise at every step
- Involves Stochastic Differential Equation (SDE)



Diffusion Models

Pros

- Superior generation quality
- State of the Art for generative models

Cons

- Conceptually complex
- Difficult to train for non-standard data-sets

AI generated Images

Stable Diffusion

OpenAI DALL·E 2

https://openai.com/dall-e-2

Socrative Quiz

Go to:
https://b.socrative.com/login/student/

Room Name

KHODA2293

Comparison

GANs	VAEs	NFs	Diffusion
Slow to train	Fast to train	Fast/Slow to train	Slow to train
Fast to run	Fast to run	(Can be made) fast to run	Slow —> Fast
Unstable to train	Tricky to balance multiple objectives, typically blurry images	Variable size outputs is tricky	Adapting to new data formats is tricky
Became famous for realistic images		Well suited for blobby physics problems	All the hype in ML world

Tutorial

Now let's practice what we learnt!

We will work on GAN, VAE and NormalizingFlows

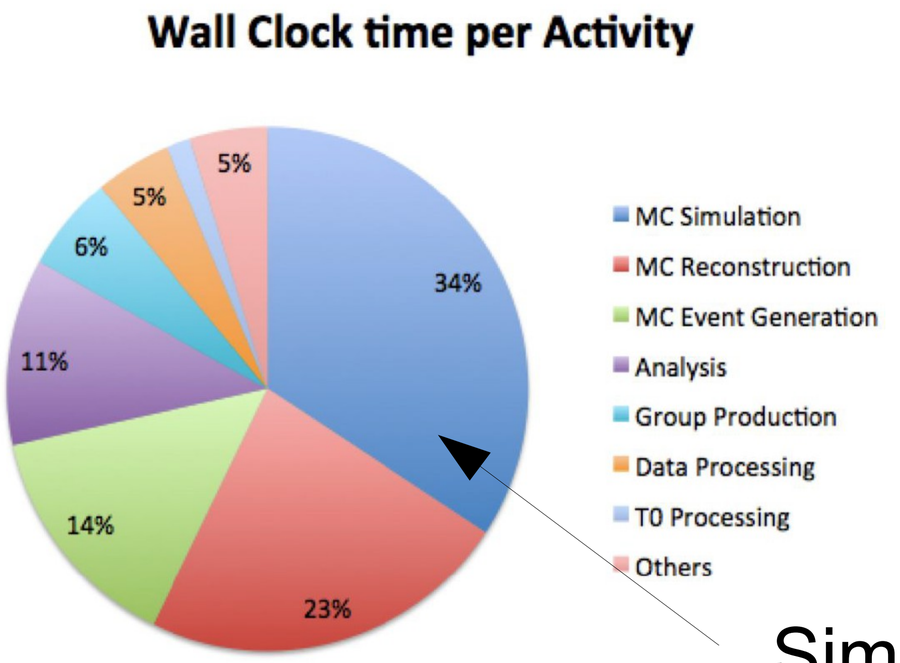
Notebook for today:
https://github.com/ml4hep-India/icts-2023/blob/main/generative_models/Flow_Tutorial_PyTorch.ipynb

You can find the solution notebook in the same git repository:
https://github.com/ml4hep-India/icts-2023/blob/main/generative_models/Flow_Tutorial_PyTorch_Solutions.ipynb

Applications

Generative Models

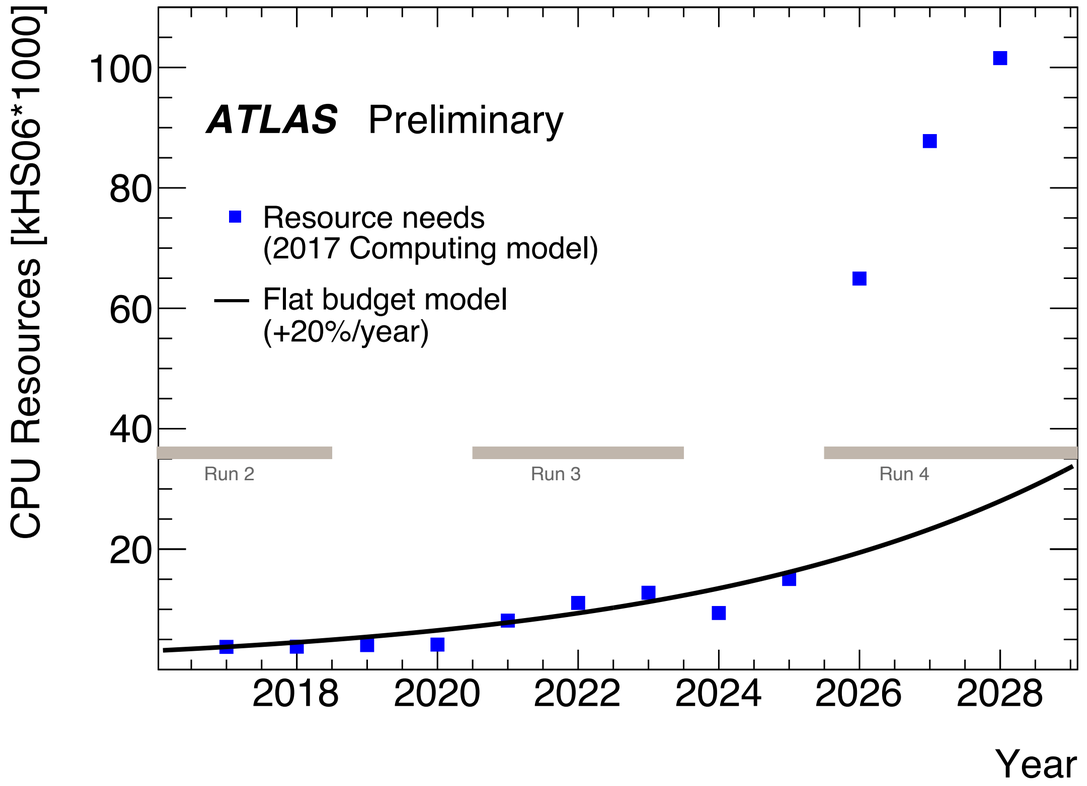
Demand >> Supply !



ATLAS 2016 numbers

Simulation!

(~75% of that for calorimeter sim alone)

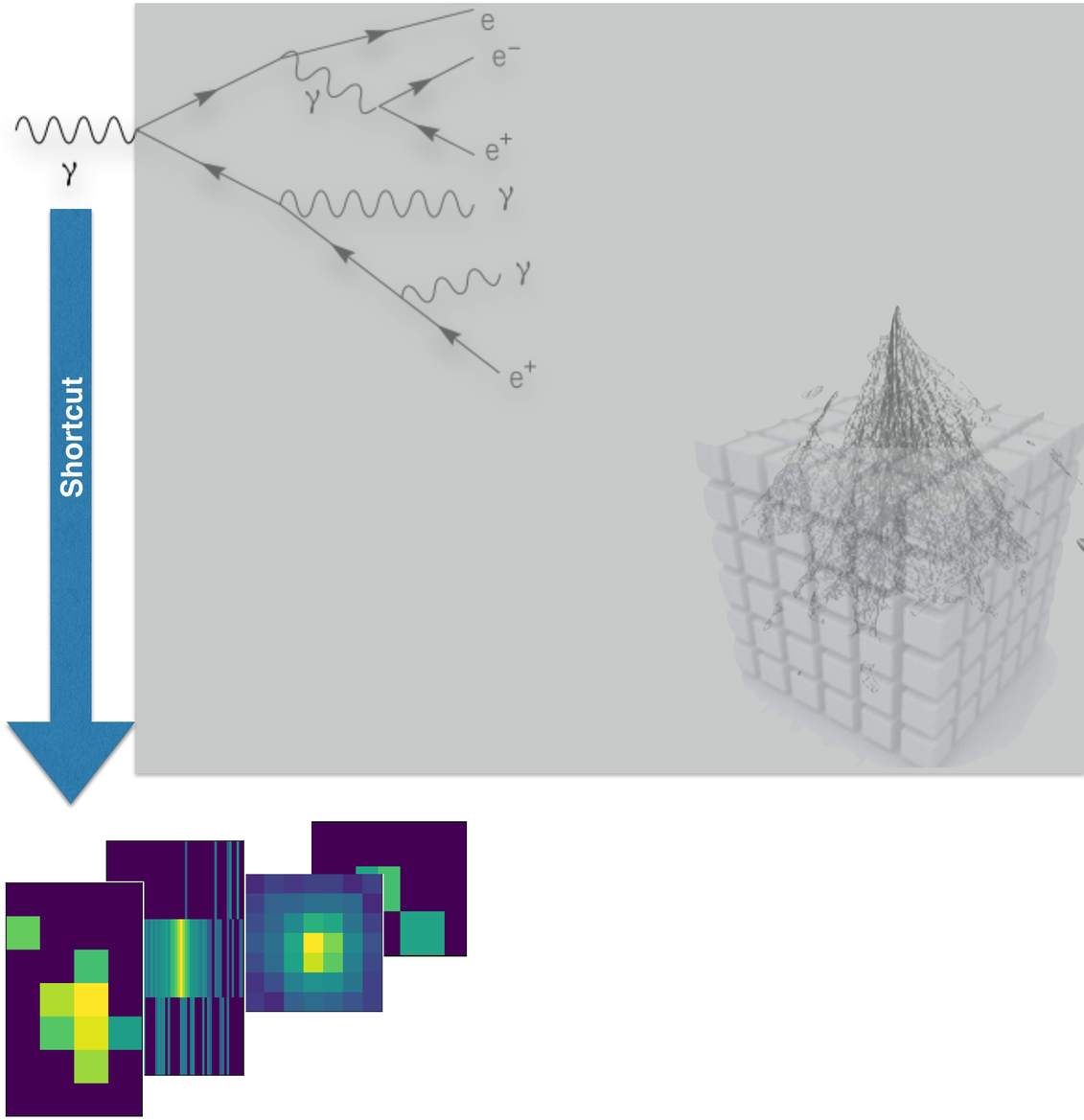


Source

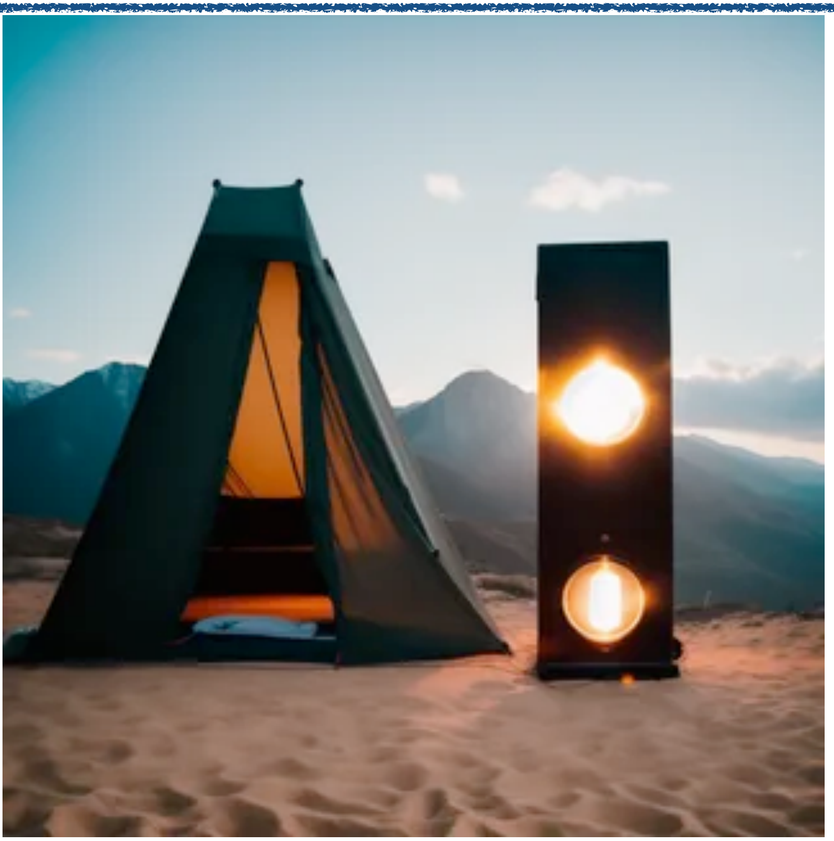
Calorimeter Response

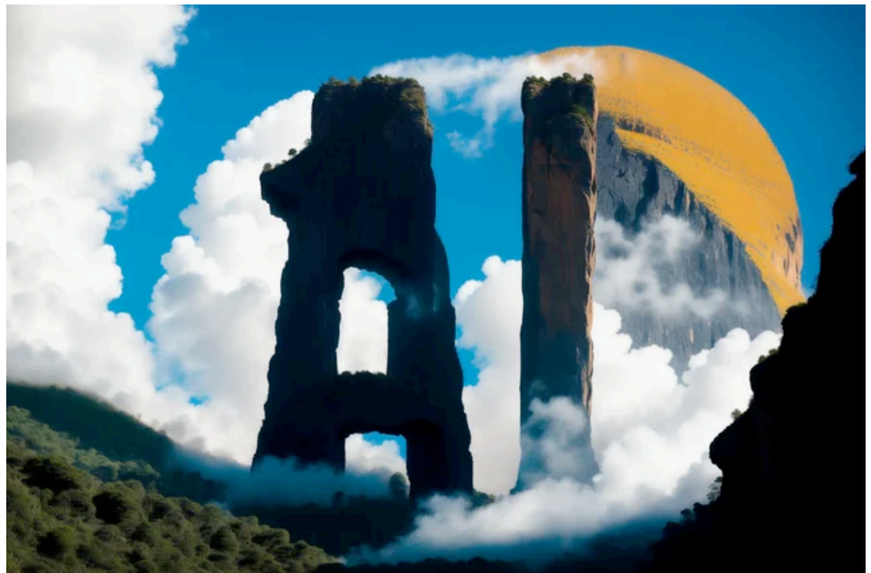
Only final image recorded

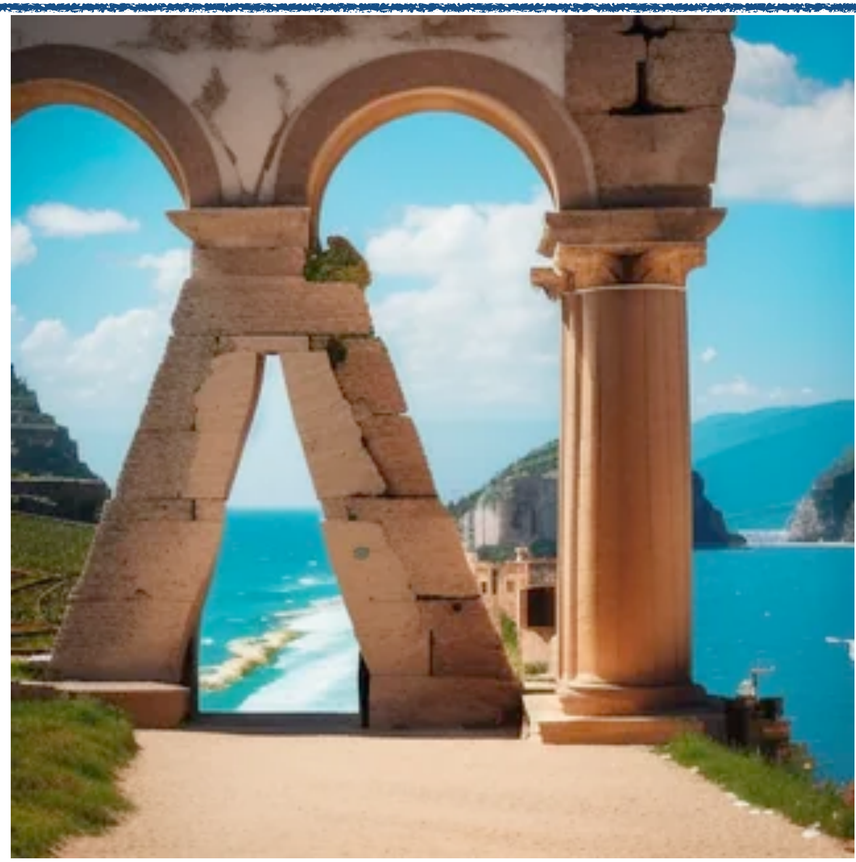
Why track full time evolution of all particles ?



Generative Models





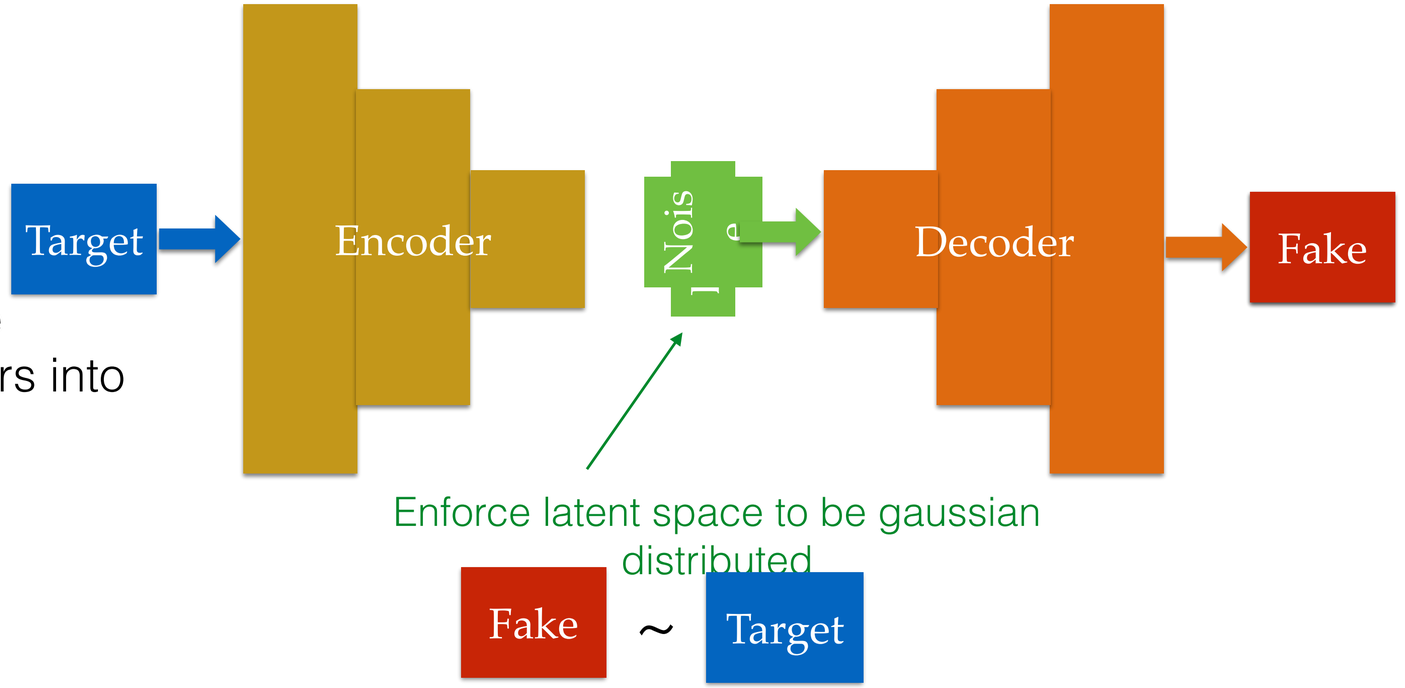


Images: Source
(StableDifusion)

First Wave of Deep Generative Models

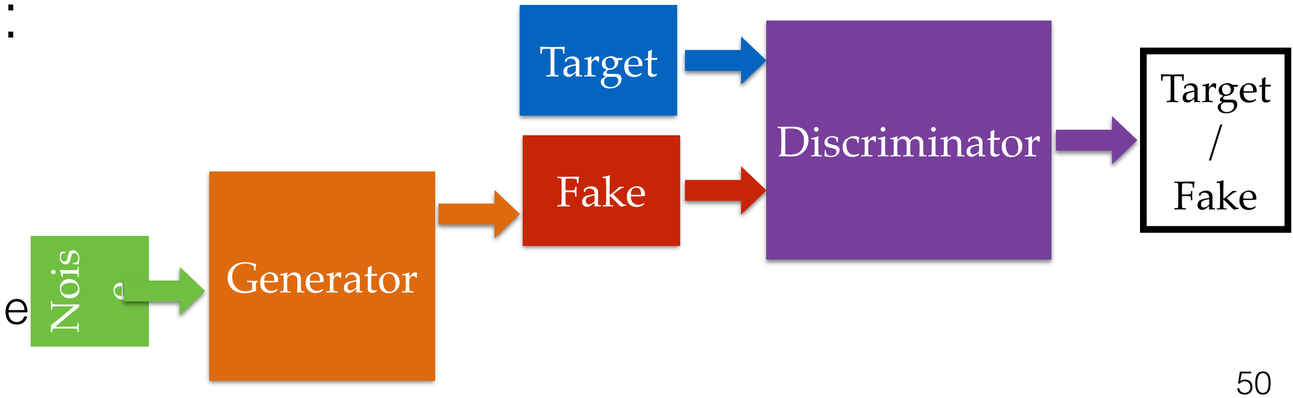
Variational AutoEncoder (VAE):

- Train encoder and decoder neural networks
- Small (often Gaussuan) encoded latent space
- Once trained, inject Gaussian random numbers into decoder to get new images



Generative Adversarial Network (GAN):

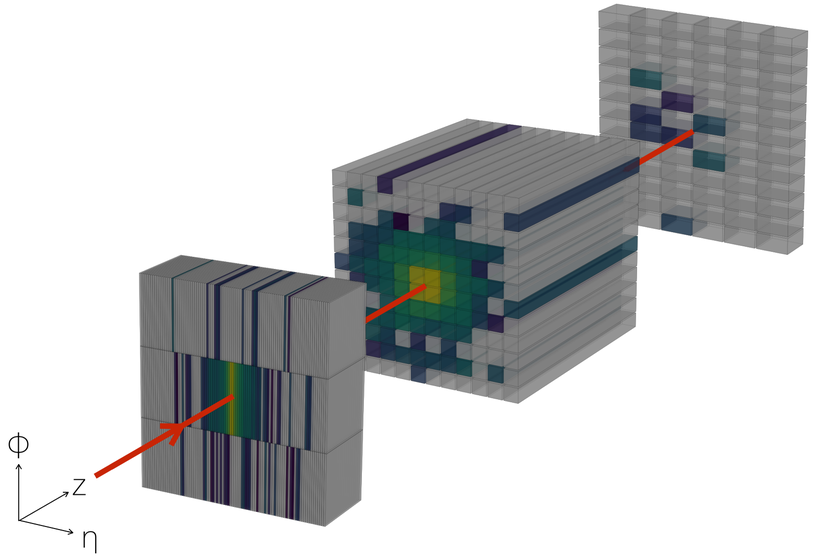
- Train a discriminative network to learn the difference between real and fake images
- Train a generative network to produce realistic fake images, to fool the discriminator (iterative)
- If converged, generator produces very realistic image



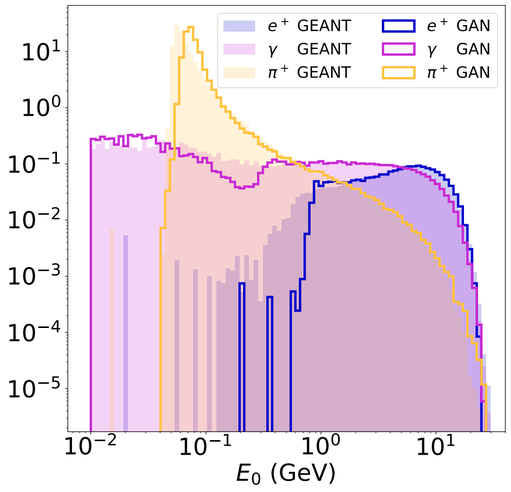
The paper that started it all

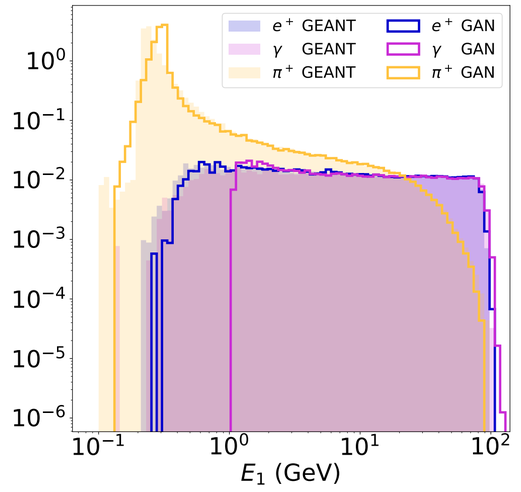
CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks

Paganini et al. (2018)



- GAN trained to reproduce ‘Geant’
- Proof of concept on a toy calorimeter



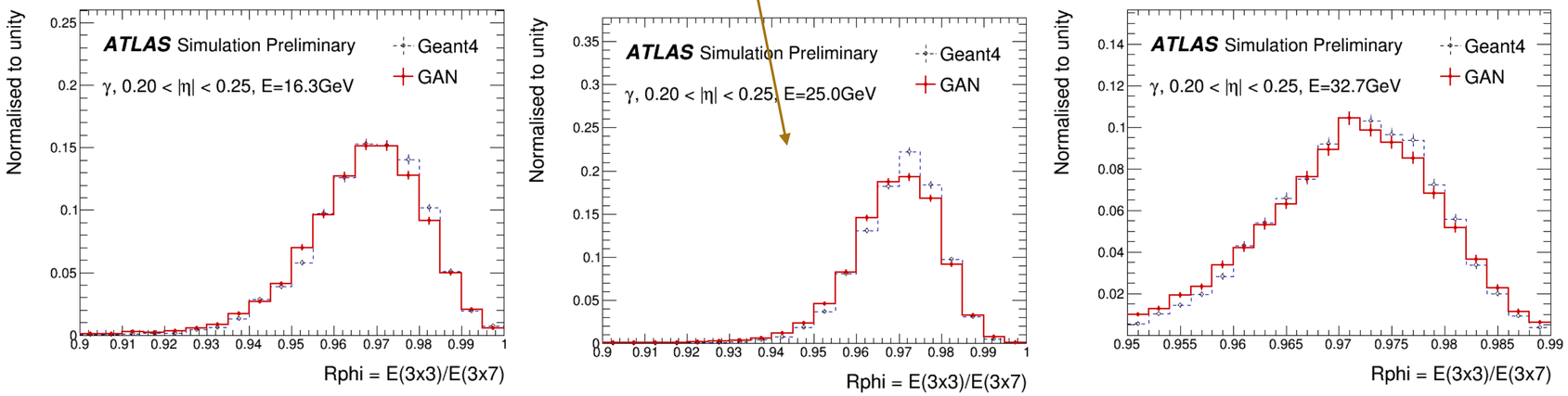


ATLAS Experiment Calorimeter Simulation (2019)

First to be integrated into full C++ simulation infrastructure

- Trained GAN & VAE on **cropped 3D images in central region**
- Validated in ATLAS software using high level physics variables
- **Interpolation test:** Train on 9 log-spaced energy points, test on 10th
- **Speed:** Orders of magnitude faster than Geant4, no longer bottleneck in sim time
- **Memory:** Tiny memory footprint O(MBs) compared to O(GBs) for baseline

GAN never trained at 25 GeV!



16 GeV | 25 GeV | 32 GeV

Details here here
VAE updates here

Challenge: Condition on detector geometry

Alignment of cell edges in 3D changes from image to image

Cell sizes change from region to region in the detector

Conditioning on $4 \times 2 = 8$ configurations already challenging, cannot scale to full detector

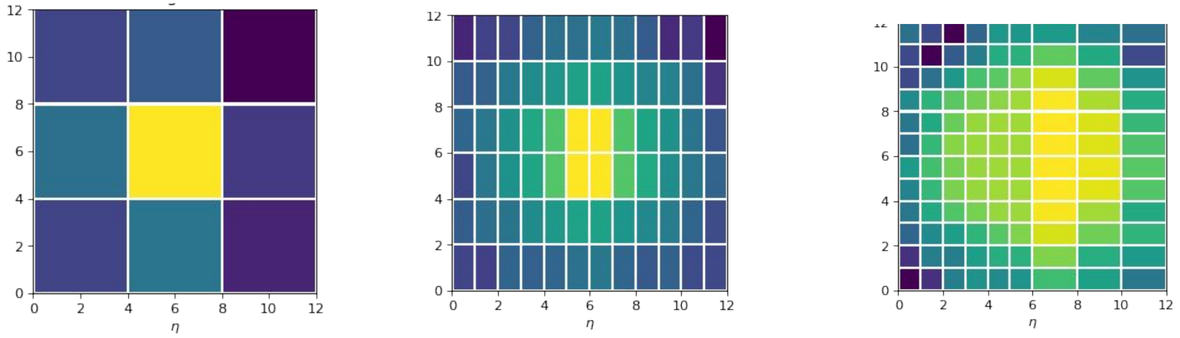
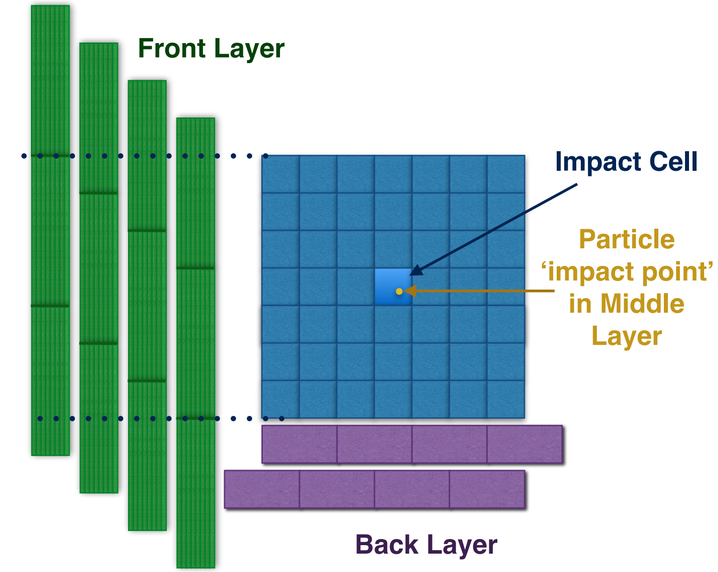
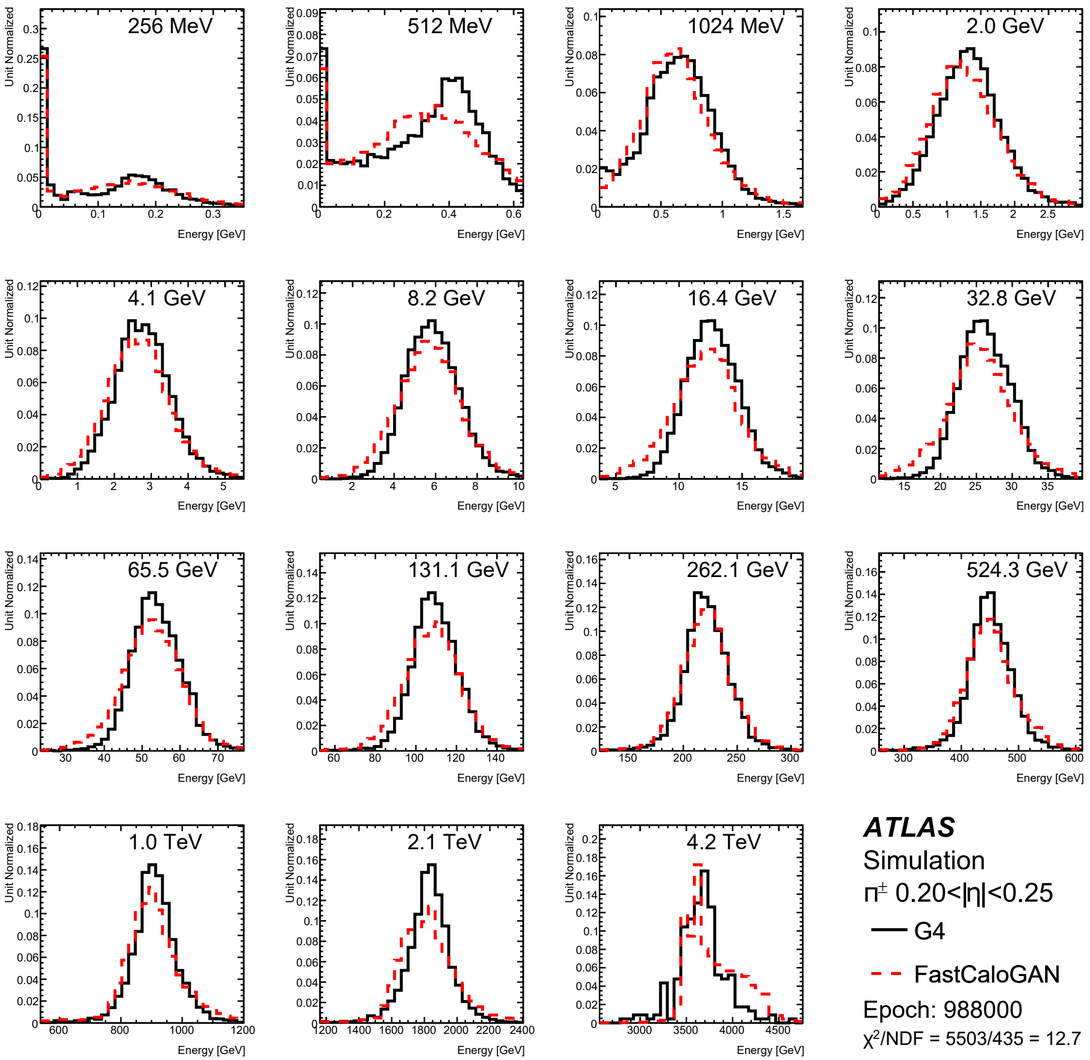


Image: Source

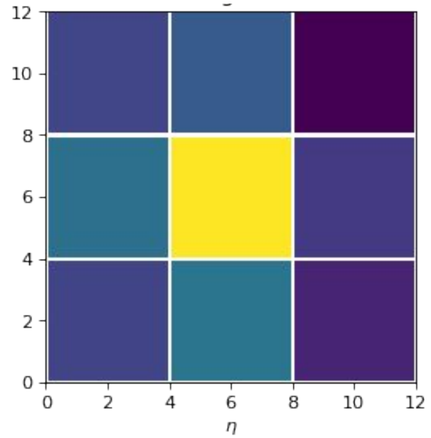


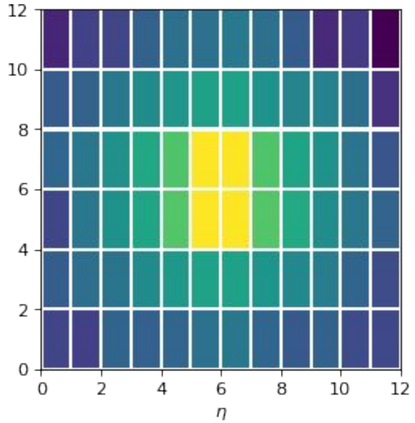
Train 100s of GAN?

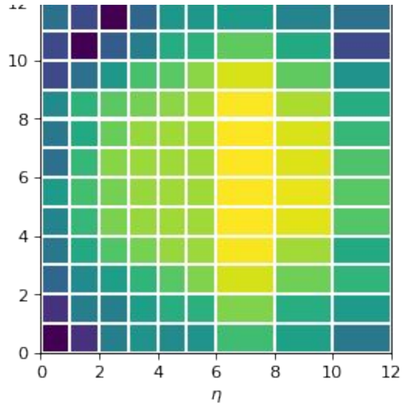


ATLASFast3 paper, 2022

A common approach to multi-geometry calo sim ?



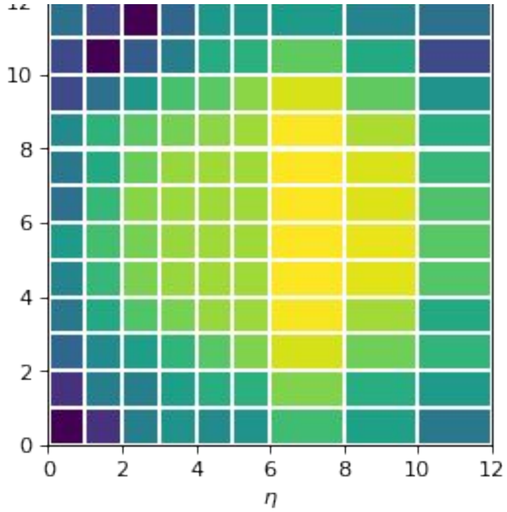


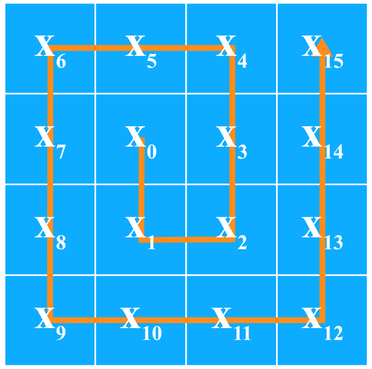


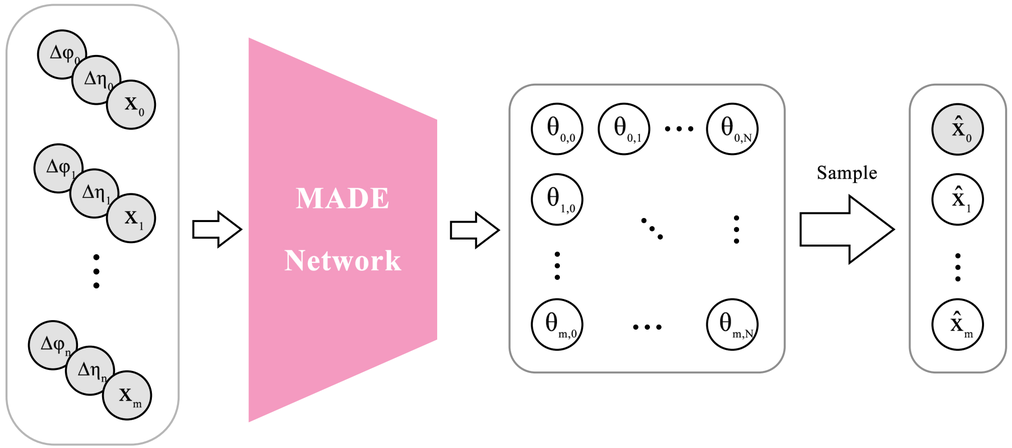
The crucial idea: Can we condition and interpolate over cell geometries ? Let the model learn concepts of ‘cell size’, ‘cell position’

Geometry-aware Autoregressive Models for Calorimeter Shower Simulations: Junze Liu, **Aishik Ghosh**, Dylan Smith, Pierre Baldi, Daniel Whiteson

Geometry Aware Learning !



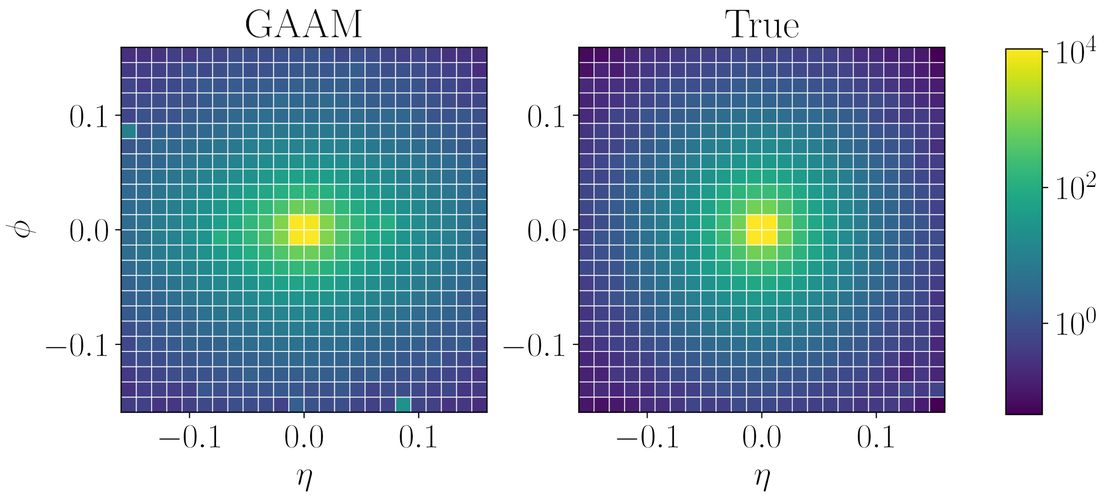




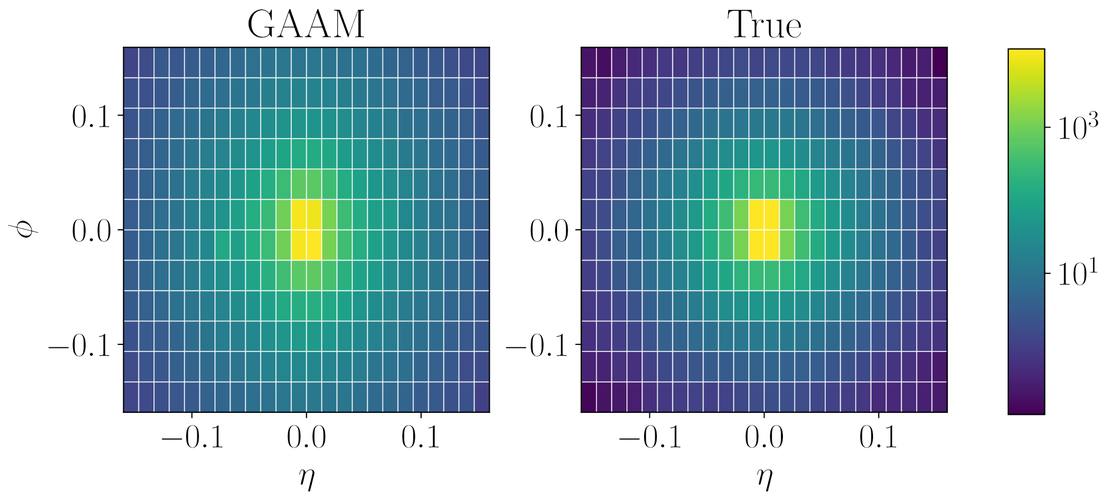
Auto-regressive model generates one cell at a time, conditioned on the properties of the cell and history of simulation.

Learning to generalise over cell properties within a single image !

Generalises to unseen geometries

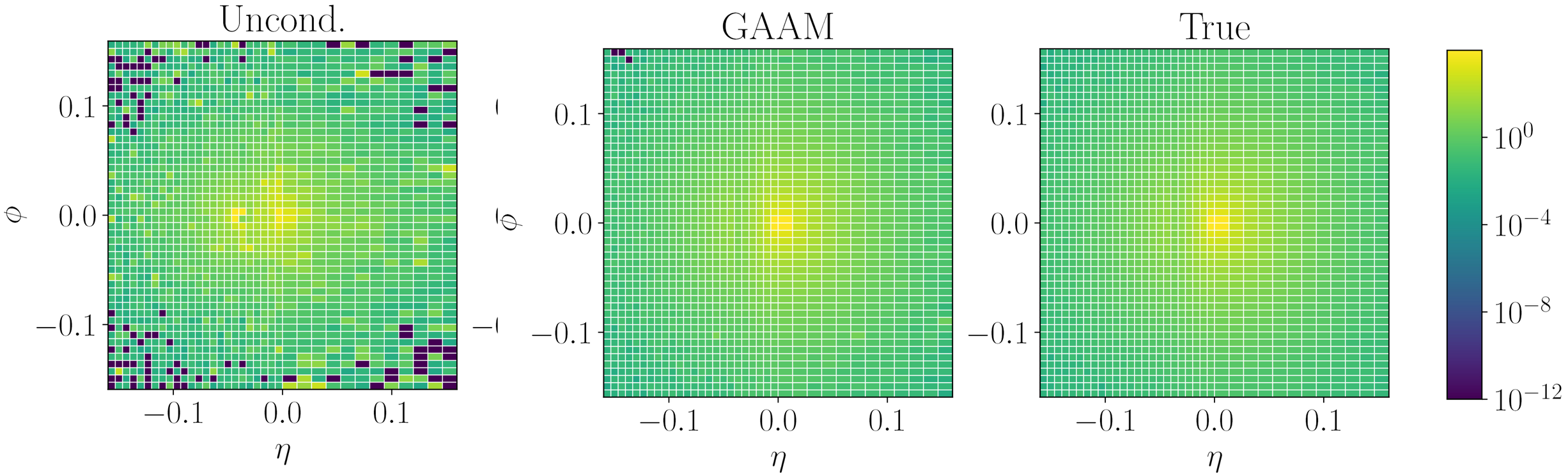


(a) Middle layer - (24, 24)



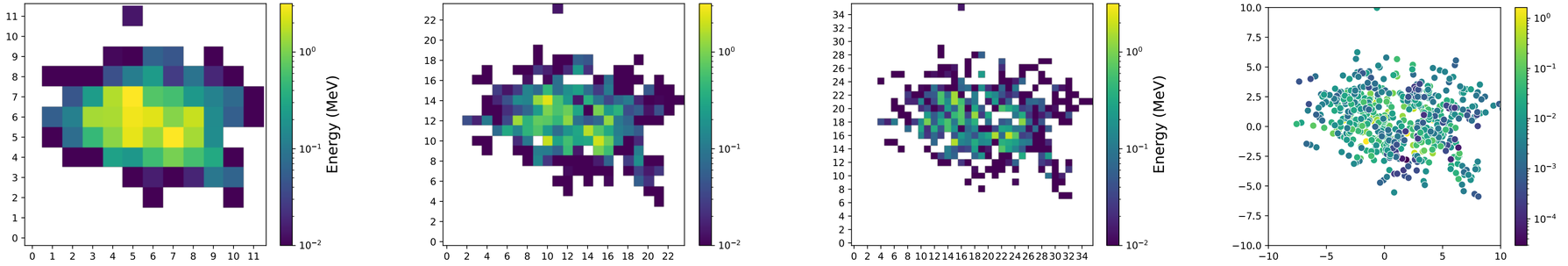
(b) Middle layer - (24, 12)

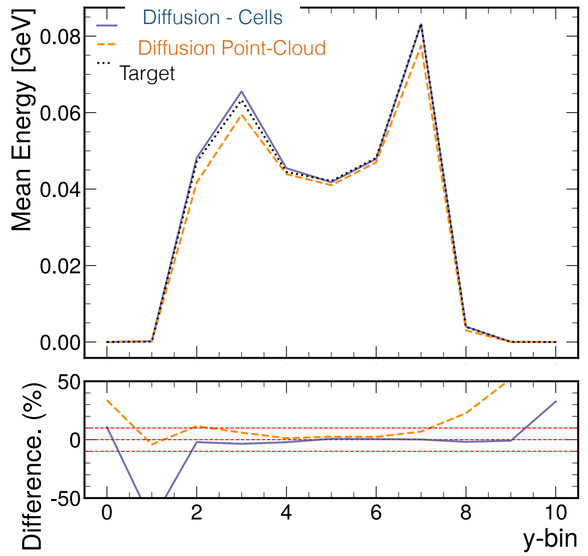
Handles boundary regions



An alternate, more challenging approach

Geometry agnostic: Simulate at maximum precision – **point-clouds** – and bin into appropriate cell sizes afterwards

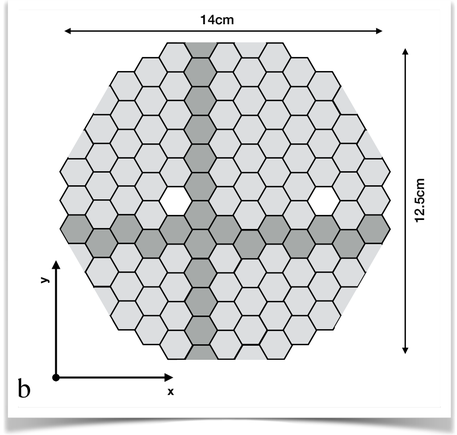




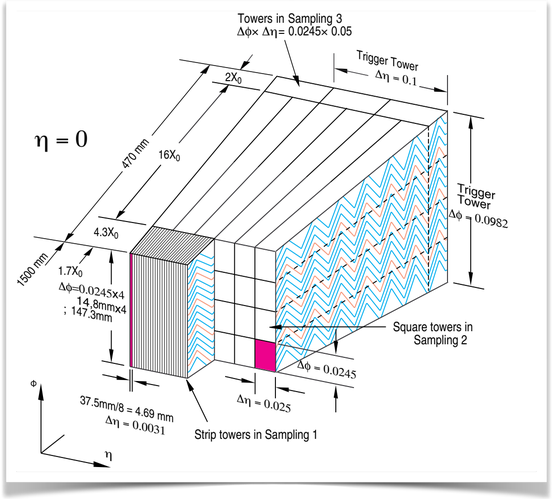
Acosta et al

Future: A common model to replace detector specific ML architectures ?

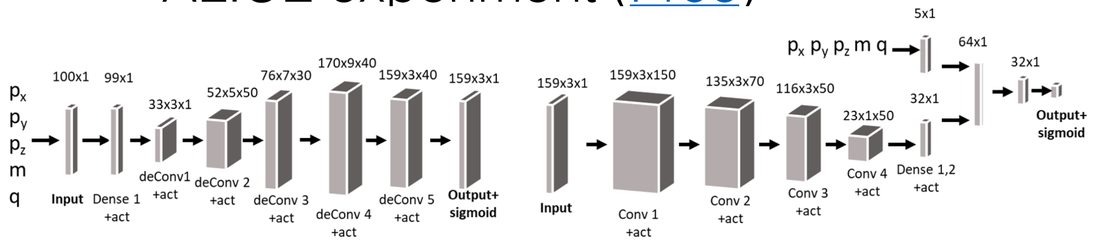
CMS experiment (Erdmann et al)



ATLAS experiment (latest paper)



ALICE experiment (Proc)



‘**Foundational Models**’: General purpose pre-trained models, to be quickly fine-tuned to your own calorimeter

- Release pre-trained models under some public license
- Experiments save years of R&D

Issues with off-the-shelf ML models

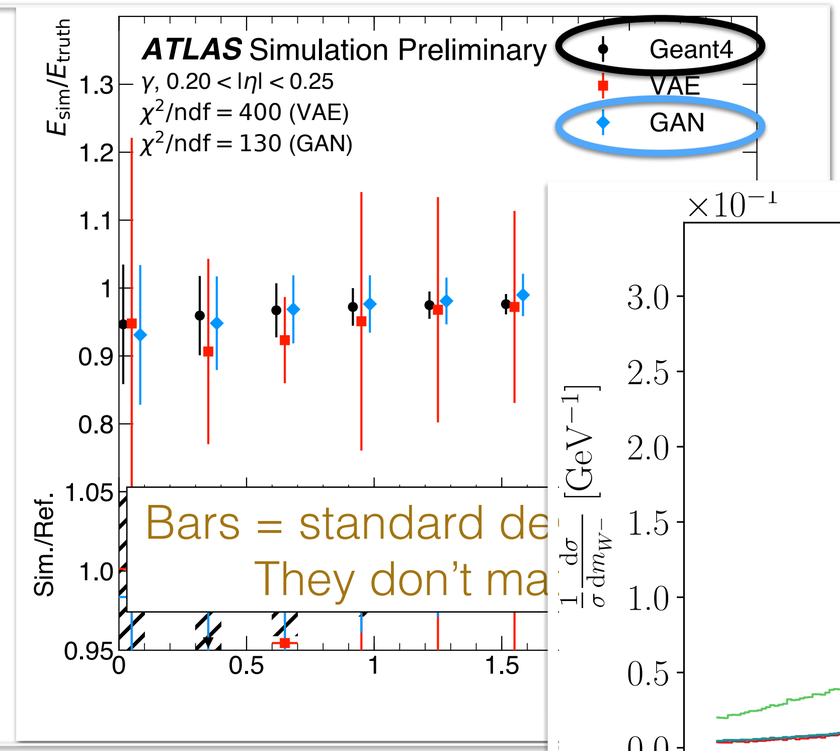
ML world not focused on modelling entire distributions

BEGAN paper

We observe varied poses, expressions, genders, skin colors, light exposure, and facial hair. However we did not see glasses, we see few older people and there are more women than men. For comparison

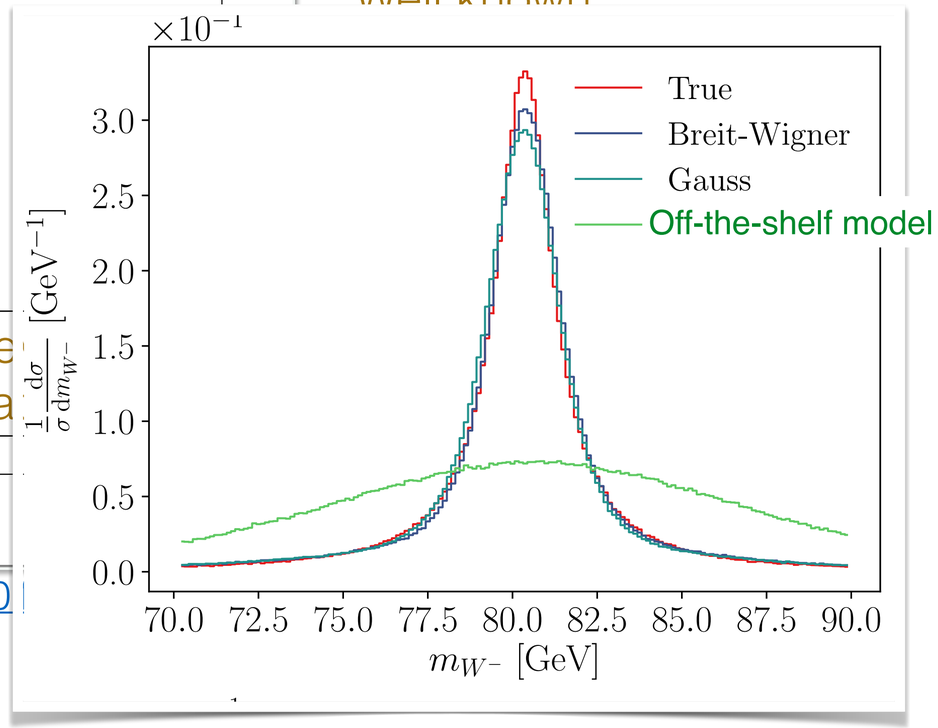
Pixel Intensities

Pixel intensities ↔ Energy per calo cell

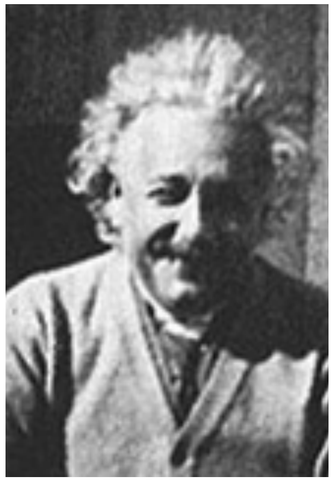


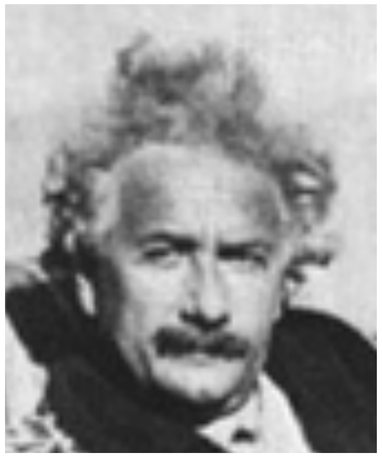
ATL-SOFT-PUB-2018-00

Well known



Butter et al.





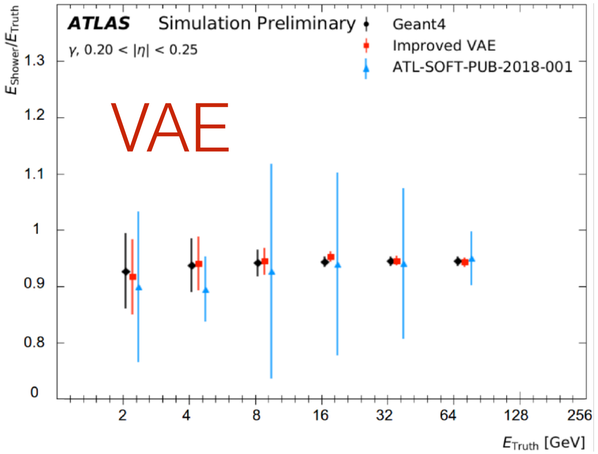
Raw pixel intensities not important for computer vision, very important for calorimetry

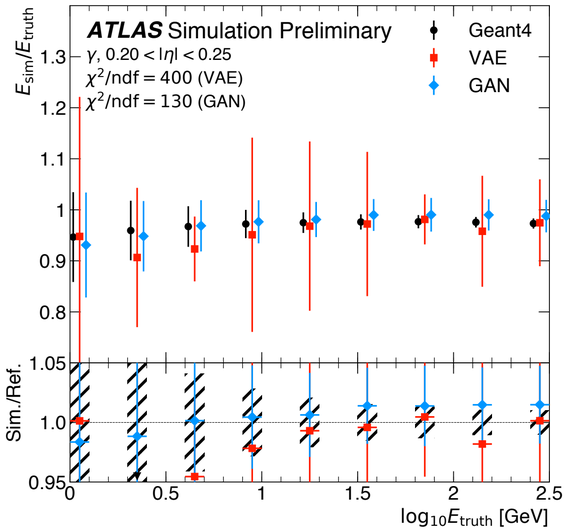
From Ben Nachman's talk, image

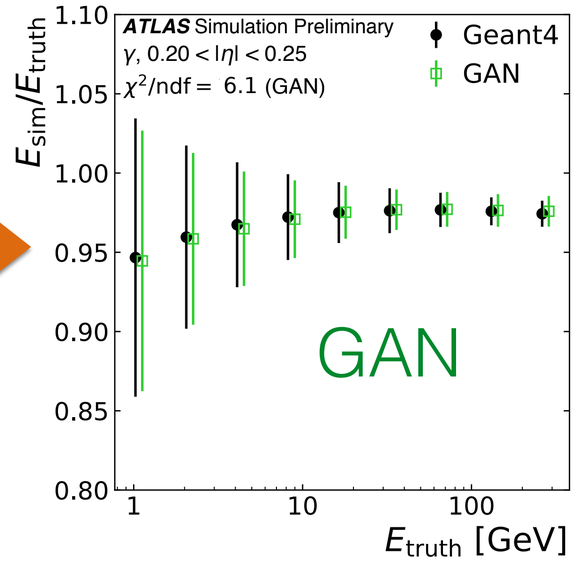
Handling pixel intensities

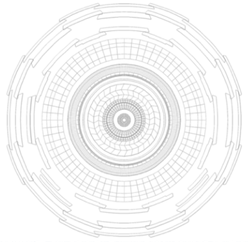
Details here here
VAE updates here

- Novel two critic WGAN architecture
- Normalisation of calo layer energies
- Or separately generate total energy per layer

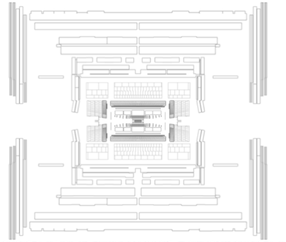








Why not re-use the training data ?



Generative models:

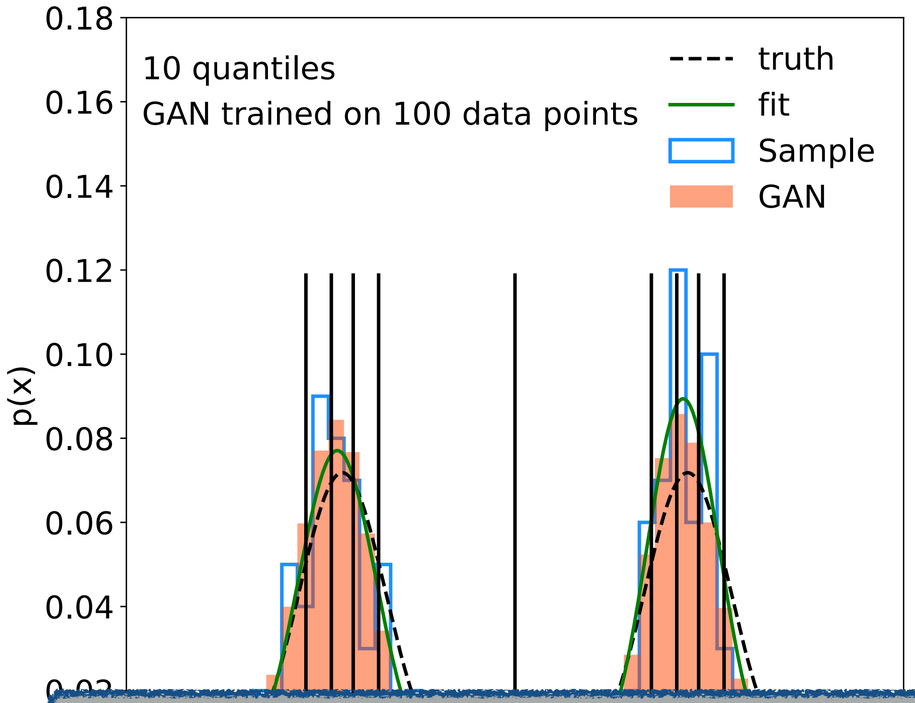
1. More memory efficient, think of NNs are an efficient compression
2. Interpolation between training points
3. Self-regularised nature of NNs let’s you smooth over statistical fluctuations in training set

‘GANplification’

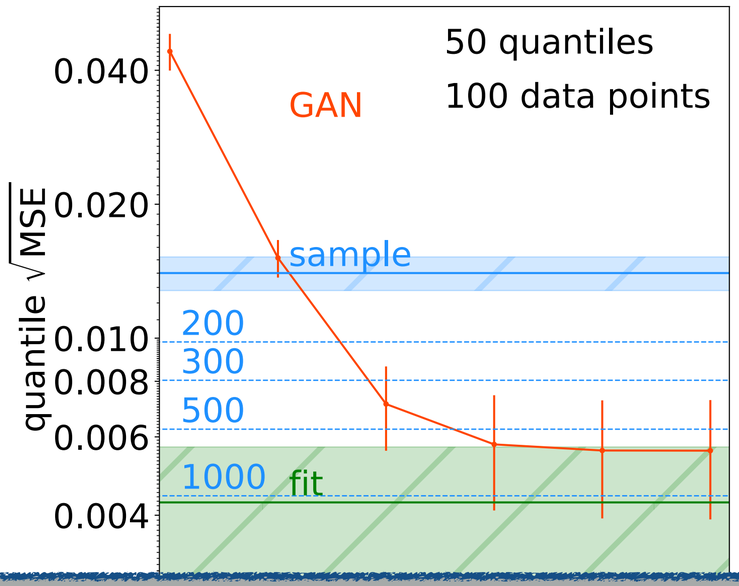
Butter, Diefenbacher et al

See more in Bieringer et al

1D camel back function



GAN effectively produces **5x** samples



Open question: How can we estimate this factor without actually having 5x samples ?

Evaluation of generative models

Evaluating Calo Simulators

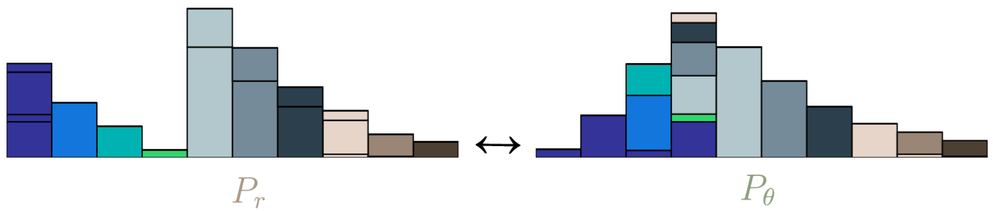
Is this the best use of human resources ?

Can we automise the evaluation ?

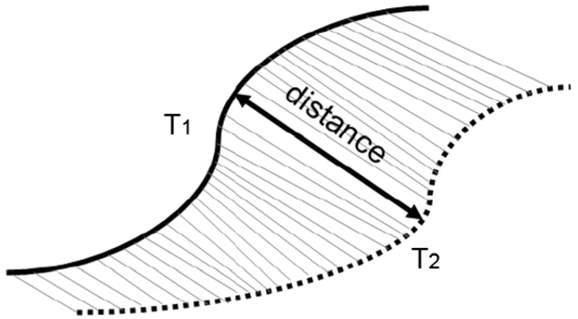
Need measures of distance → Active field of research in ML / statistics / mathematics

$$\frac{P(x \mid Geant)}{P(x \mid Gen)}$$

Likelihood Ratio



Wasserstein Distance



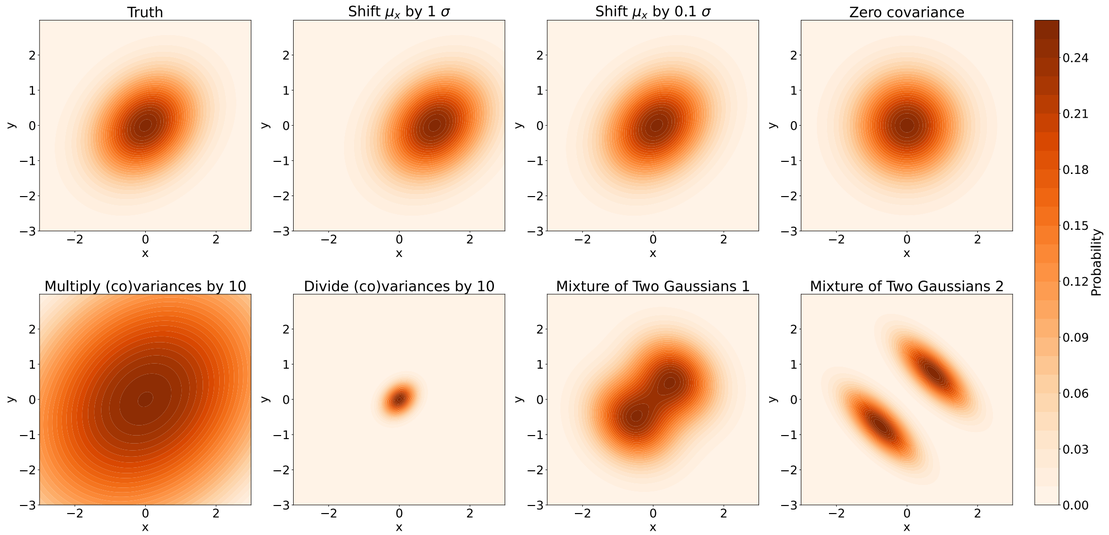
Fréchet Distance

Kansal et al, 2022

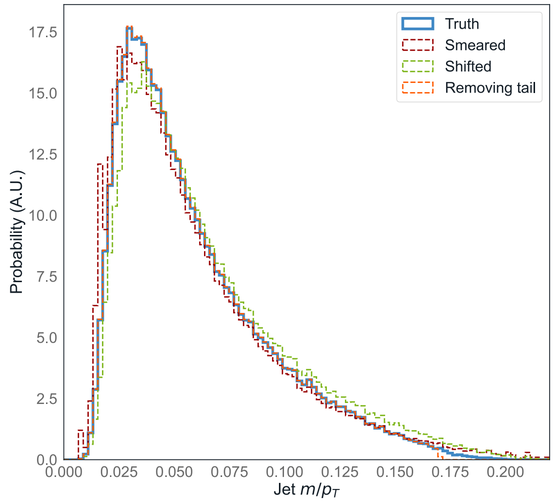
A large comparison of metrics

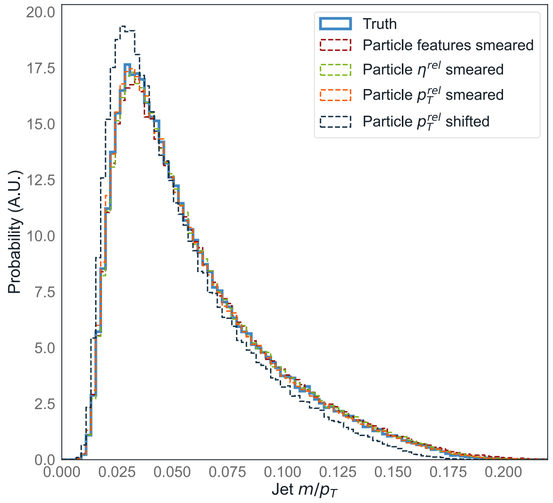
On the Evaluation of Generative Models in High Energy Physics

Test on Gaussian examples:

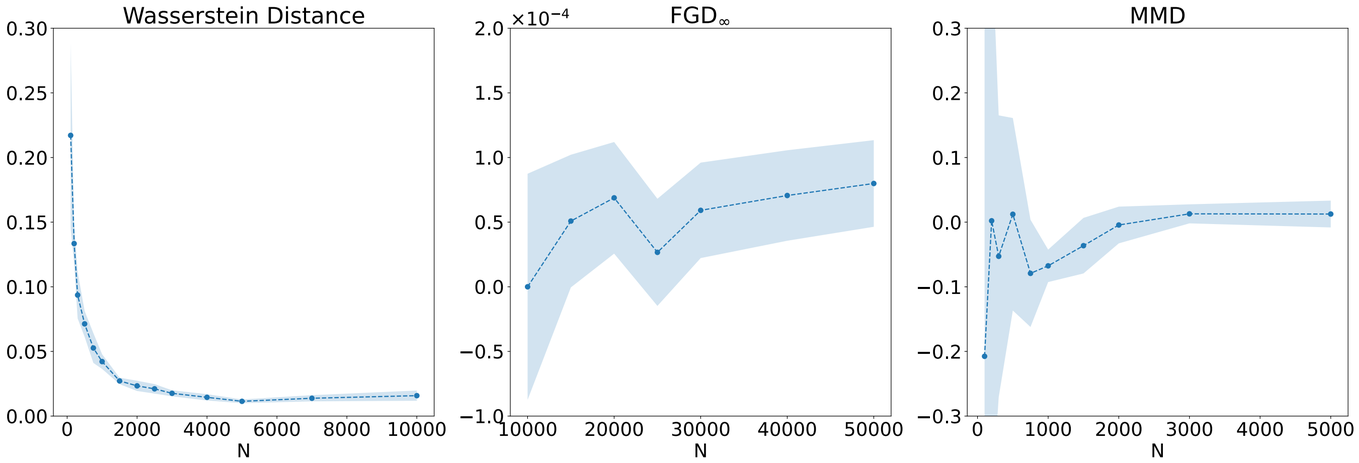


Test on Jet Physics examples:





Findings of this study



Metric	Truth	Smeared	Shifted	Removing tail	Particle features smeared	Particle η^{rel} smeared	Particle $p_{\text{T}}^{\text{rel}}$ smeared	Particle $p_{\text{T}}^{\text{rel}}$ shifted
$W_1^M \times 10^3$	0.28 ± 0.05	2.1 ± 0.2	6.0 ± 0.3	0.6 ± 0.2	1.7 ± 0.2	0.9 ± 0.3	0.5 ± 0.2	5.8 ± 0.2
Wasserstein EFP	0.02 ± 0.01	0.09 ± 0.05	0.10 ± 0.02	0.016 ± 0.007	0.19 ± 0.08	0.03 ± 0.01	0.03 ± 0.02	0.06 ± 0.02
FGD_∞ EFP $\times 10^3$	0.01 ± 0.02	$\mathbf{21.5 \pm 0.3}$	$\mathbf{26.8 \pm 0.3}$	$\mathbf{2.31 \pm 0.07}$	23.4 ± 0.3	$\mathbf{3.59 \pm 0.09}$	2.29 ± 0.05	28.9 ± 0.2
MMD EFP $\times 10^3$	-0.006 ± 0.005	0.17 ± 0.06	0.9 ± 0.1	0.03 ± 0.02	0.35 ± 0.09	0.08 ± 0.05	0.01 ± 0.02	1.8 ± 0.1
Precision EFP	0.9 ± 0.1	0.94 ± 0.04	0.978 ± 0.005	0.88 ± 0.08	0.7 ± 0.1	0.94 ± 0.06	0.7 ± 0.1	0.79 ± 0.09
Recall EFP	0.9 ± 0.1	0.88 ± 0.07	0.97 ± 0.01	0.92 ± 0.06	0.83 ± 0.05	0.92 ± 0.07	0.8 ± 0.1	0.8 ± 0.1
Wasserstein PN	1.65 ± 0.06	1.7 ± 0.1	2.4 ± 0.4	1.71 ± 0.08	4.5 ± 0.1	1.79 ± 0.05	4.0 ± 0.4	7.6 ± 0.2
FGD_∞ PN $\times 10^3$	0.8 ± 0.7	40 ± 2	193 ± 9	5.0 ± 0.9	$\mathbf{1250 \pm 10}$	20 ± 1	$\mathbf{1230 \pm 10}$	$\mathbf{3640 \pm 10}$
MMD PN $\times 10^3$	-2 ± 2	4 ± 8	80 ± 10	-1 ± 4	500 ± 100	3 ± 2	560 ± 60	1100 ± 40
Precision PN	0.68 ± 0.07	0.64 ± 0.04	0.71 ± 0.06	0.73 ± 0.03	0.09 ± 0.04	0.75 ± 0.08	0.08 ± 0.04	0.39 ± 0.08
Recall PN	0.70 ± 0.05	0.61 ± 0.04	0.61 ± 0.08	0.73 ± 0.06	0.014 ± 0.009	0.7 ± 0.1	0.01 ± 0.01	0.57 ± 0.09
Classifier LLF AUC	0.50	0.52	0.54	0.50	0.97	0.81	0.93	0.99
Classifier HLF AUC	0.50	0.53	0.55	0.50	0.84	0.64	0.74	0.92

- FGD_∞, MMD unbiased
- W too expensive for large N

FGD_∞ most promising (with caveats)

Anomaly Detection

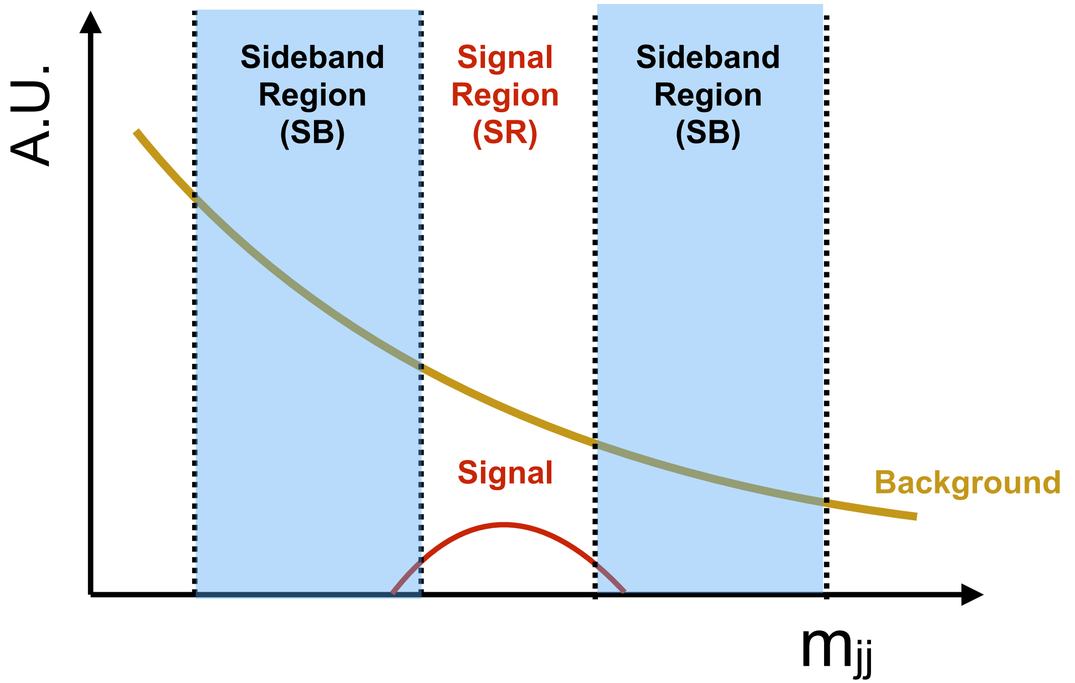
Collider Physics

Search for Resonant Signal

Assumption:

Signal is localized at least in one of the feature spaces

- Should appear as a bump



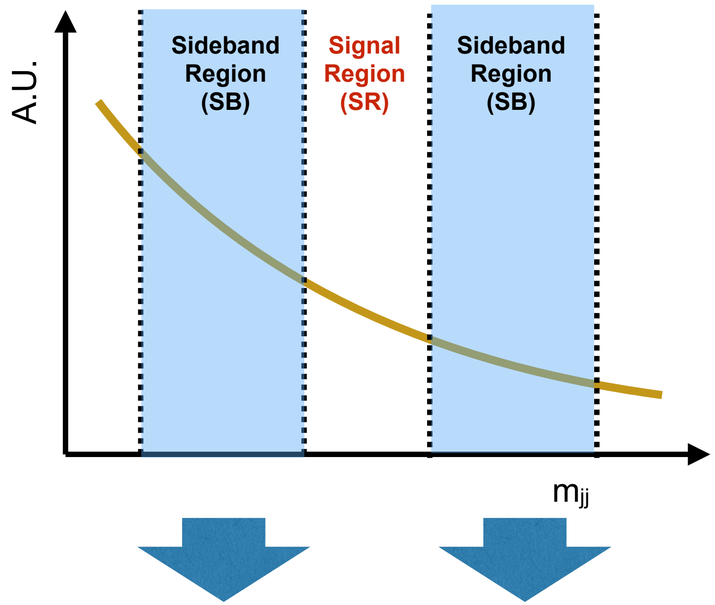
General search strategy (without ML):

- Choose a discriminant observable (often the mass)
- Define sideband regions → low signal contamination
- Fit background in the sidebands
- Interpolate the fit to the signal region

Relies only on one observable!

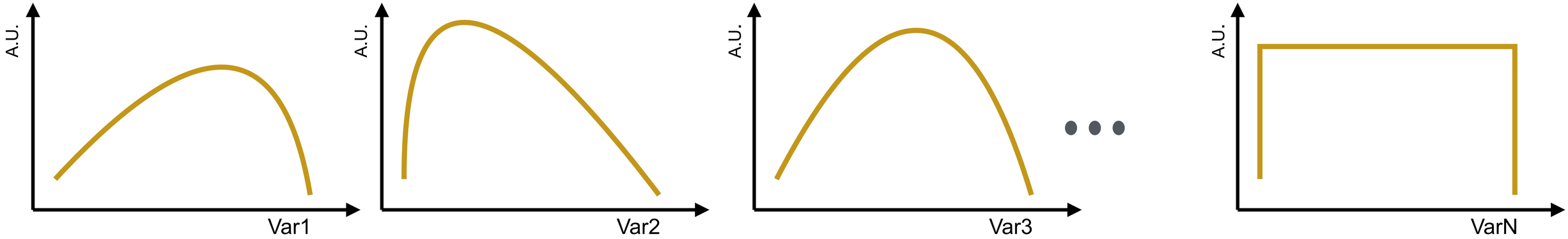
- Increase sensitivity by taking a multi-variate approach

Model background features in the SB



Side band is defined based on the observable where signal is expected to be resonant

Model the multiple observables in the sideband regions

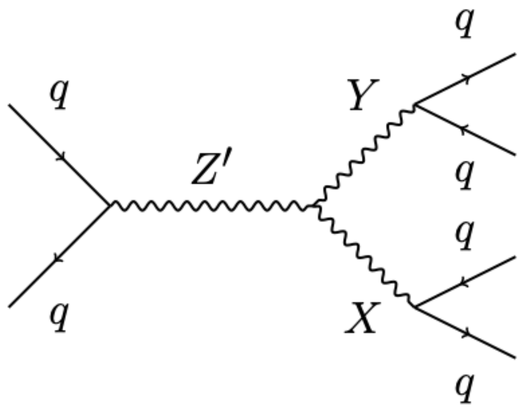


Overview of the Problem

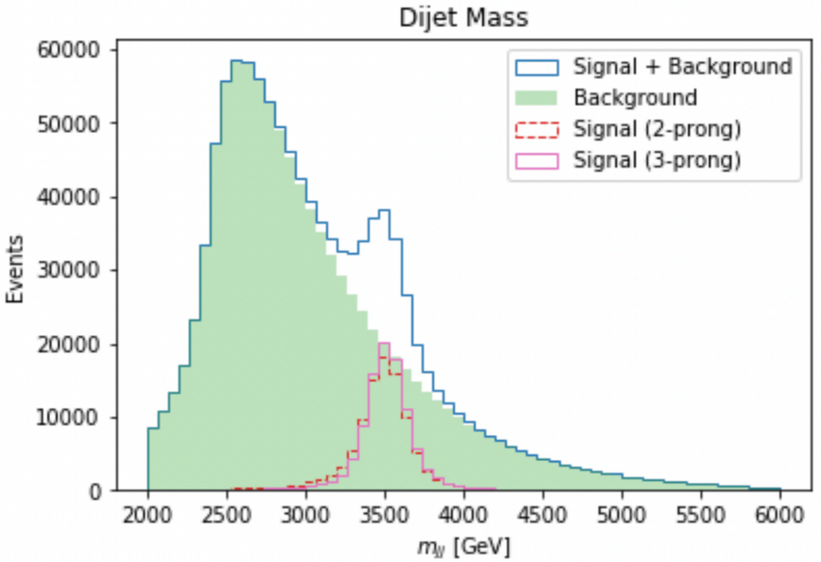
- We are working with the LHC Olympics 2020 anomaly detection challenge dataset

- **Target signature**: Final states with multiple jets

- **Background:** QCD multijet process
  - No particular structure inside the jets



- **Signal:** Heavy new particle decaying into quarks → forming large-R jets
  - with 2-prong or 3-prong structure inside (depending on the origin)

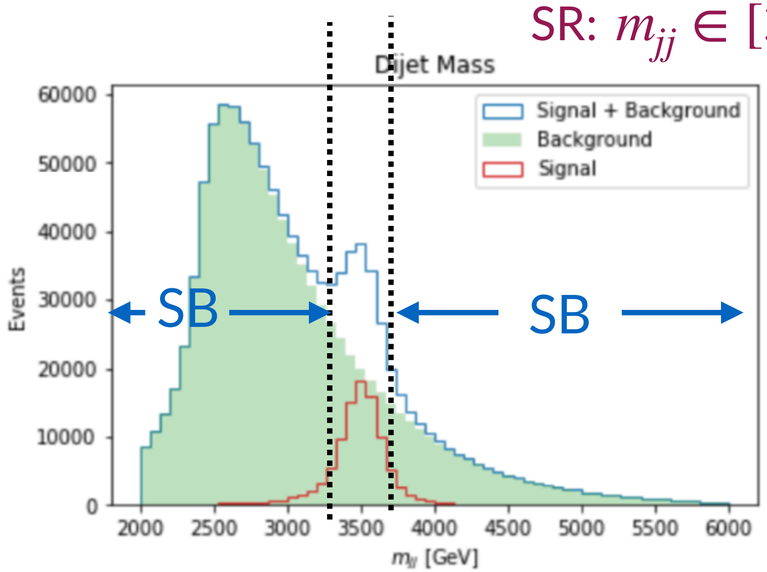


LHCO 2020 Dataset

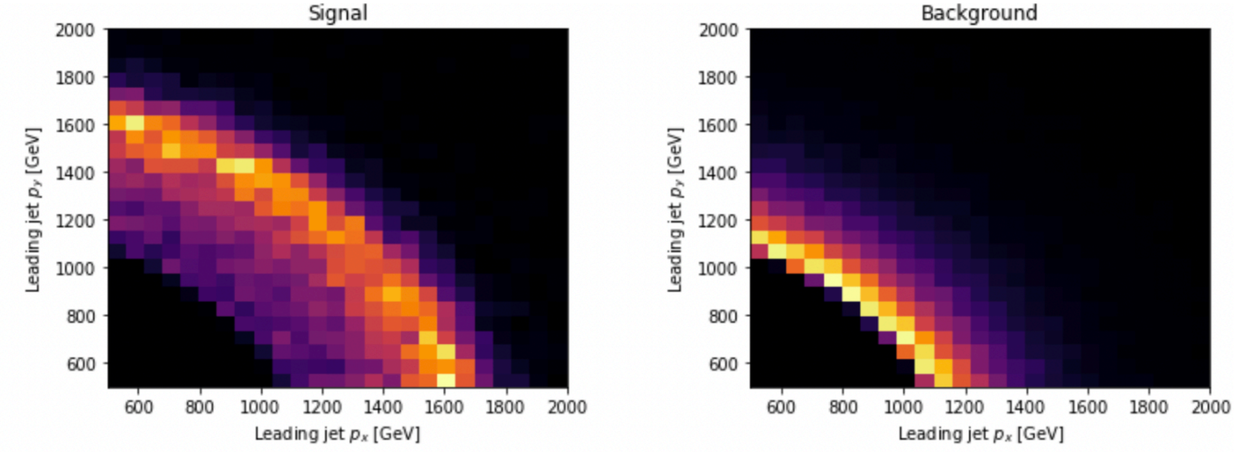
Dataset Summary:

- **Background:** 1M QCD dijet events
- **Signal:** 100k W' (3.5 TeV) → X (500 GeV) + Y (100 GeV) , with X→ qq/qqq and Y → qq/qqq
- Both **2-prong** and **3-prong** signals are used

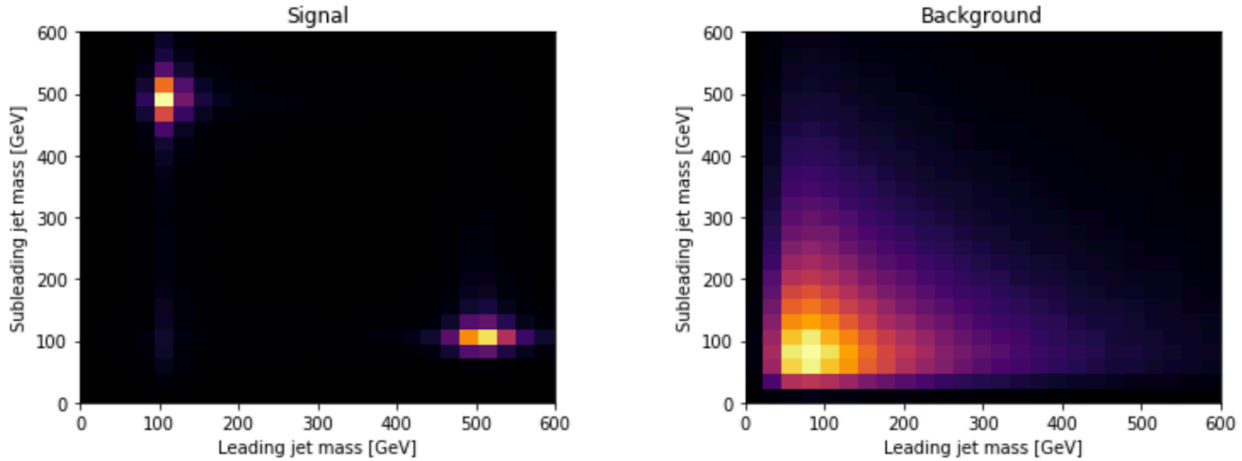
SR: $m_{jj} \in [3.3,\ 3.7]$ TeV



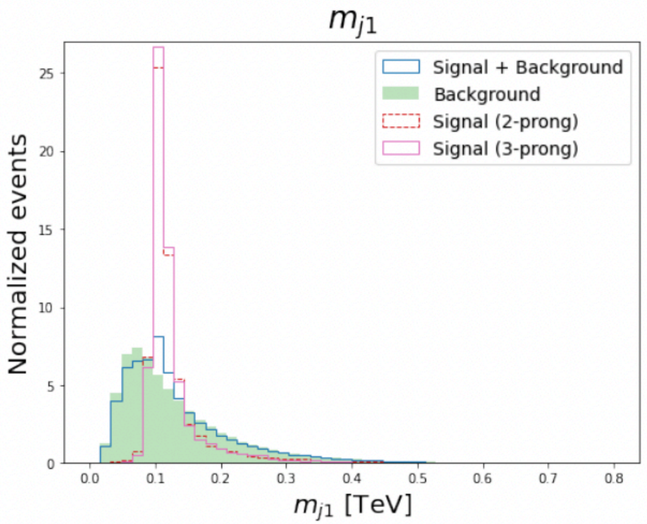
Jet Momentum

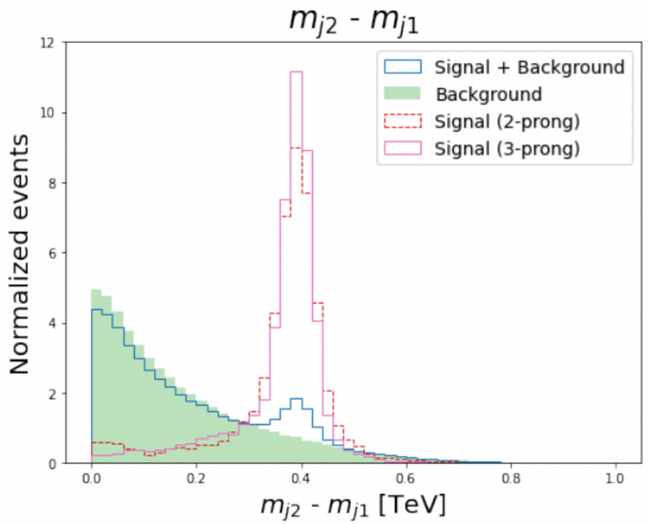


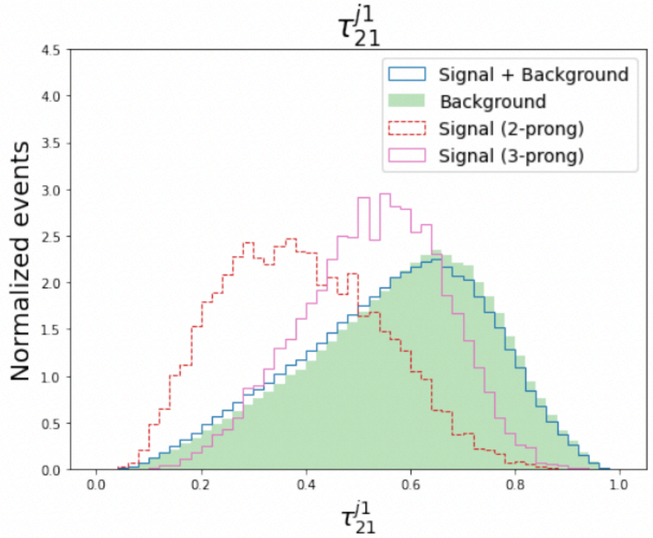
Jet Mass

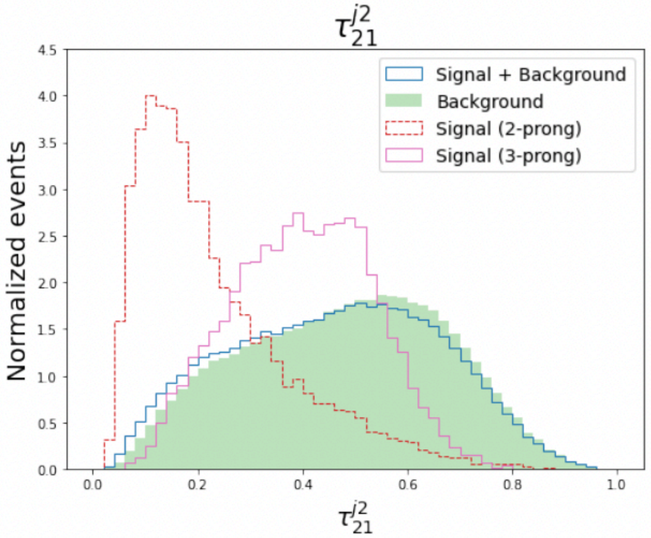


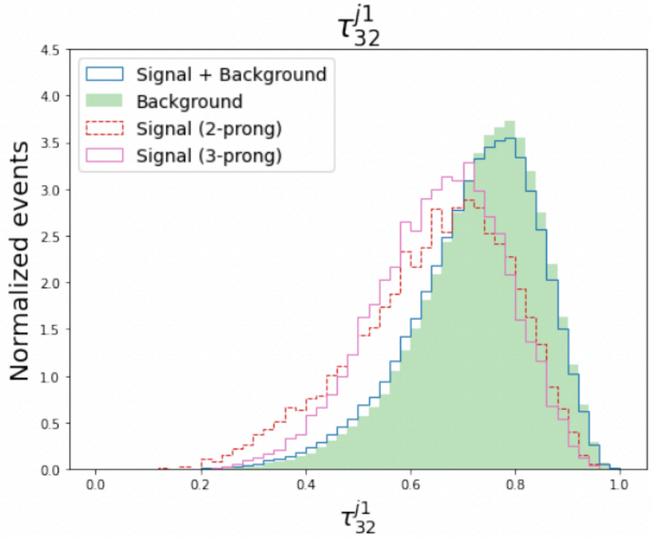
Features in the SR

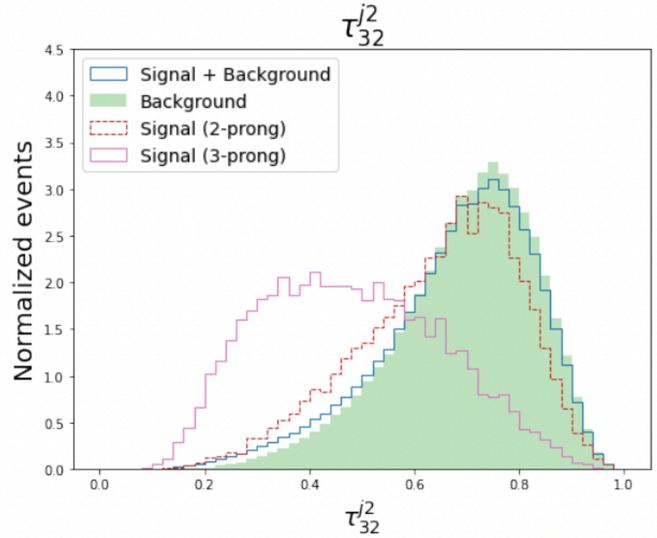












Code Practice

It is an active area of research!

Notebook section 3:
https://github.com/ml4hep-India/icts-2023/blob/main/generative_models/Flow_Tutorial_PyTorch.ipynb

- You will develop a Normalizing Flow using the LHC Olympics 2020 challenge dataset

  See the published result using this approach (CATHODE)
  https://journals.aps.org/prd/abstract/10.1103/PhysRevD.106.055006

You can see different methods originated from the challenge:
https://arxiv.org/abs/2101.08320

Thank You!

Extra Slides