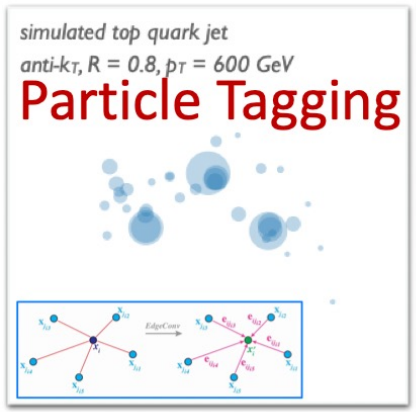
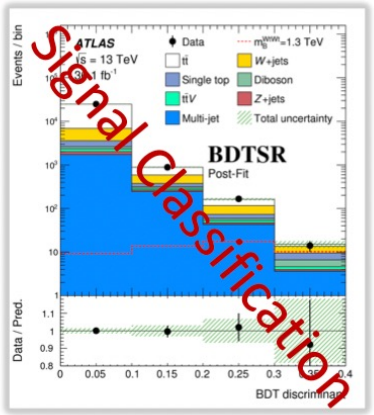
Advance machine learning techniques in HEP

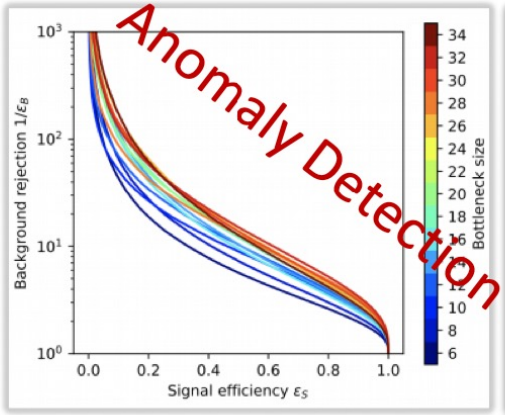
Report on CERN–Fermilab Summer School

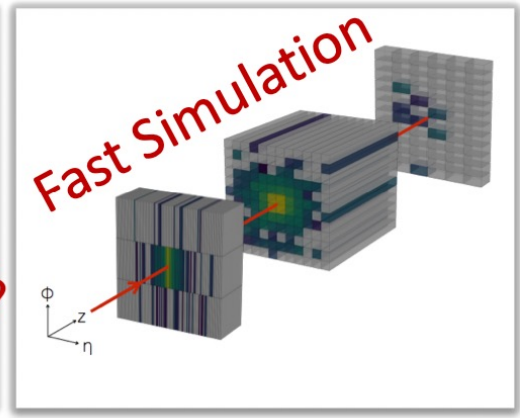
Eric Ballabene

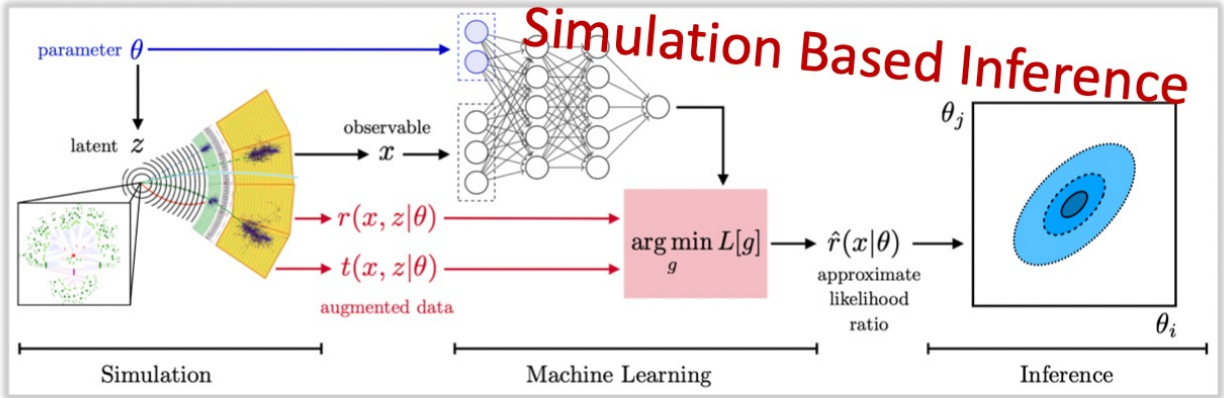
Machine Learning in HEP

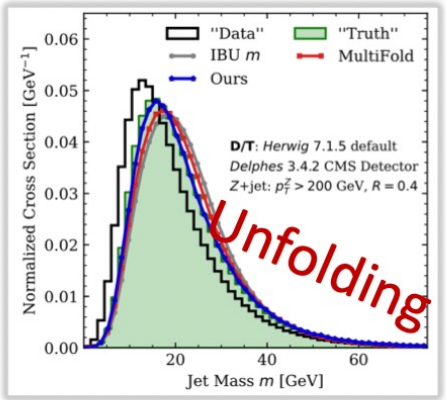


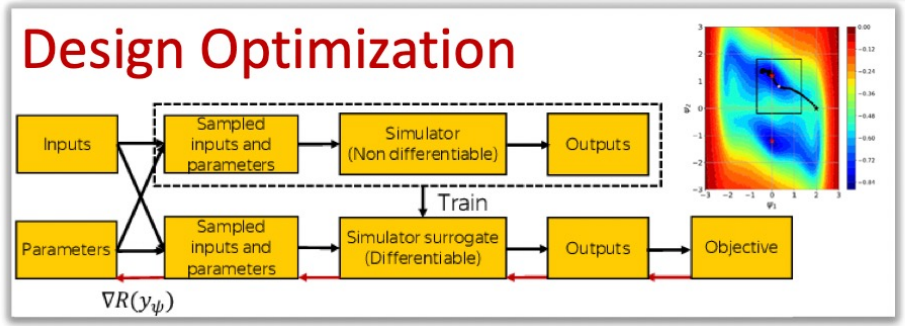


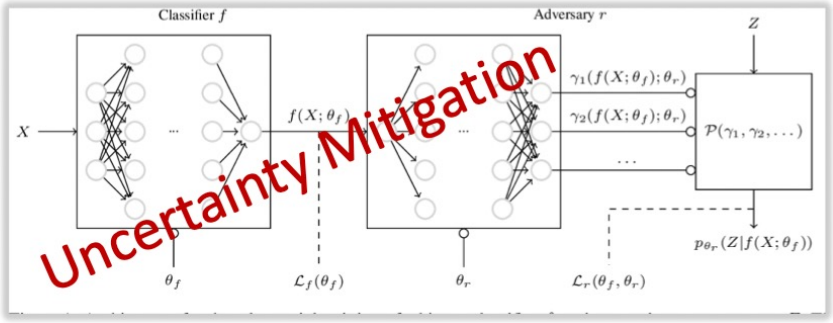










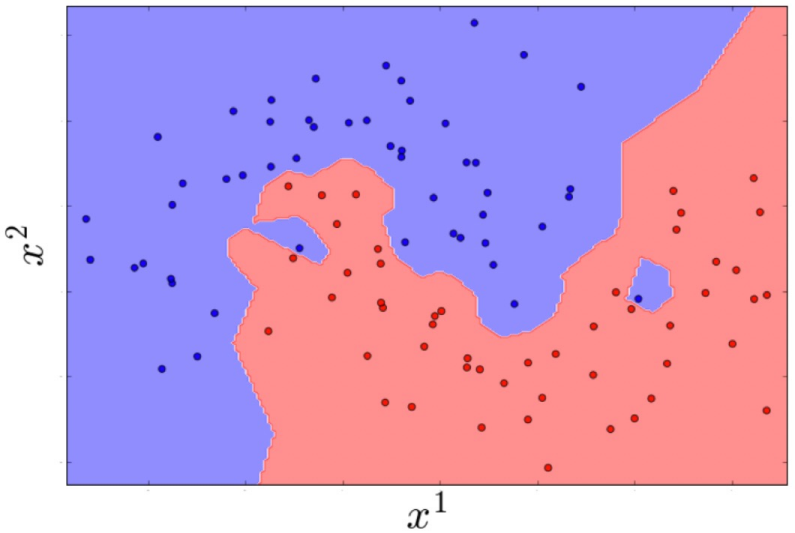


Supervised Learning

- Given N data with observable features $\{x_i \in X\}$ and prediction targets $\{y_i \in Y\}$, learn function mapping $h(x) = y$.

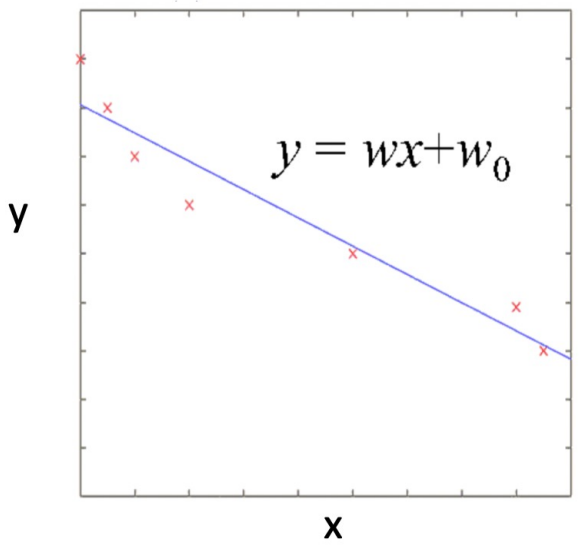
Classification

Y is a finite set of **labels** (i.e. classes) denoted with integers



Regression

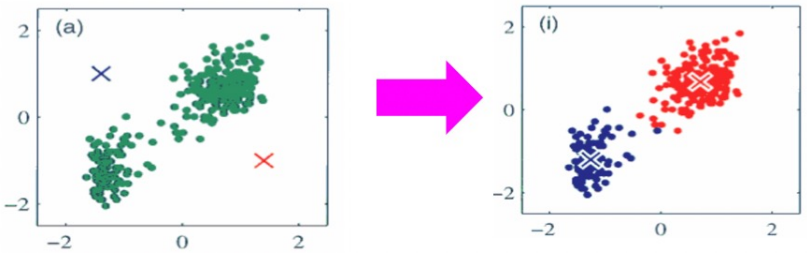
Y is a real number



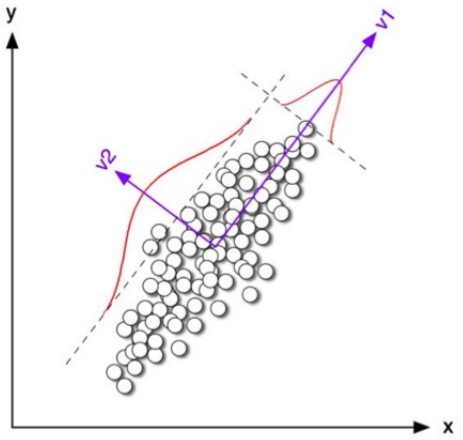
Unsupervised Learning

- Given some data $\mathrm{D} = \{x_i\}$, but no labels, find structure in data

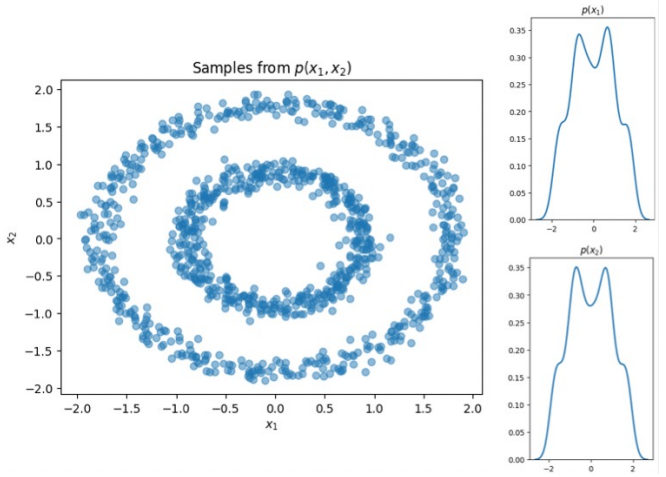
Clustering: partition the data into groups
$\mathrm{D} = \{D_1 \cup D_2 \cup \cdots \cup D_k\}$



Dimensionality reduction: find a low dimensional (less complex) representation of the data with a mapping Z=h(X)



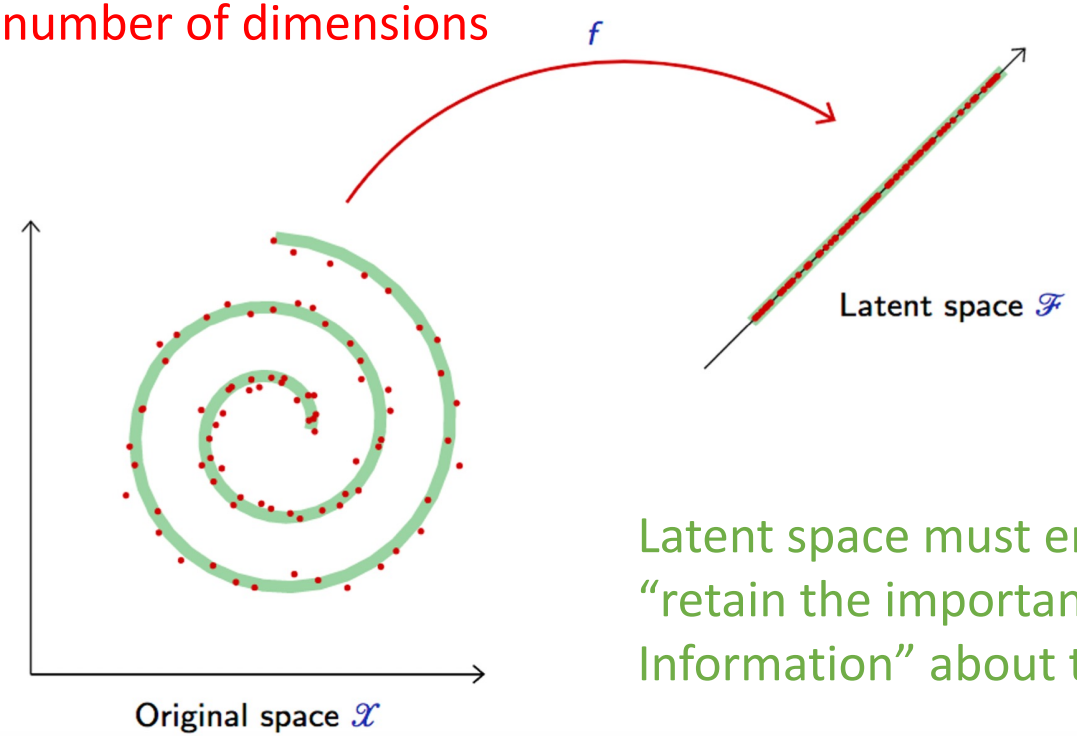
Density estimation and sampling: estimate the p(x) pdf, and/or learn to draw plausible new samples of x



Unsupervised Learning

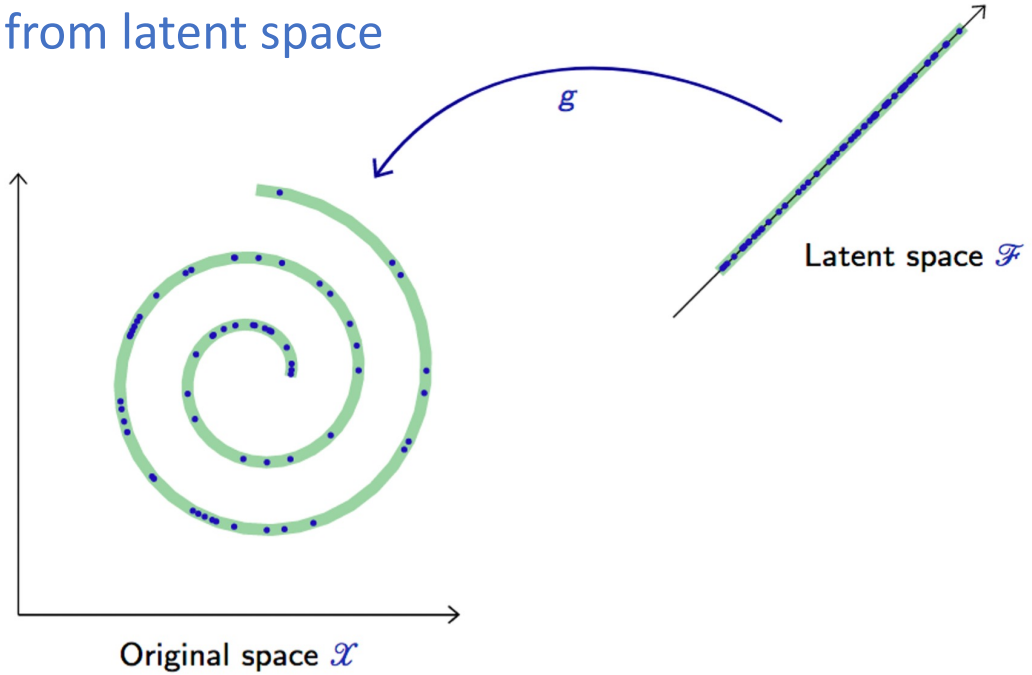
- Many tasks require **Unsupervised Learning**
- Often framed as modelling the lower dimensional "meaningful degrees of freedom" that describe the data
- Can we learn this compression and latent space?

Compress the data to a latent space with smaller number of dimensions



Latent space must encode and "retain the important Information" about the data

One way to frame "retaining important information": we can reconstruct original data from latent space



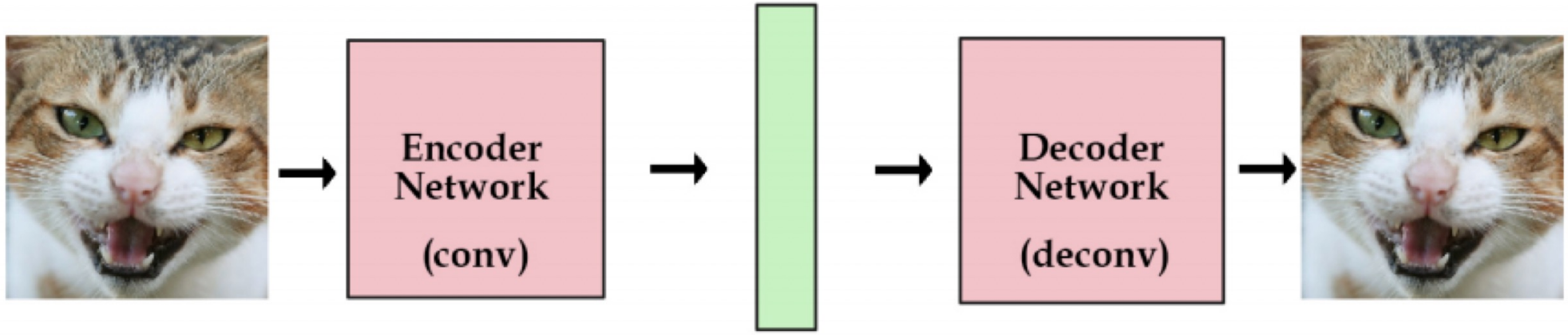
Autoencoders

Autoencoders map a space to itself through a compression

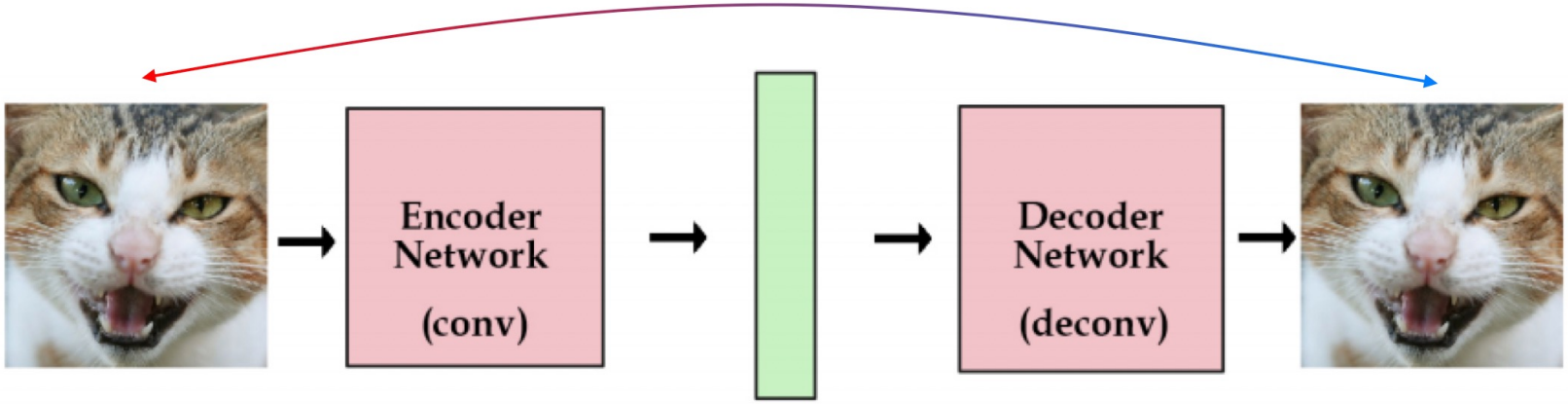
x = Data z = Latent Space

$$x \rightarrow z \rightarrow \hat{x}$$

- Full transformation should be close to the identity on the data



Autoencoders



- Full transformation should be close to the identity on the data

$$x \rightarrow z \rightarrow \hat{x}$$

$$x \rightarrow f(x) = z$$

$$z \rightarrow g(z) = \hat{x}$$

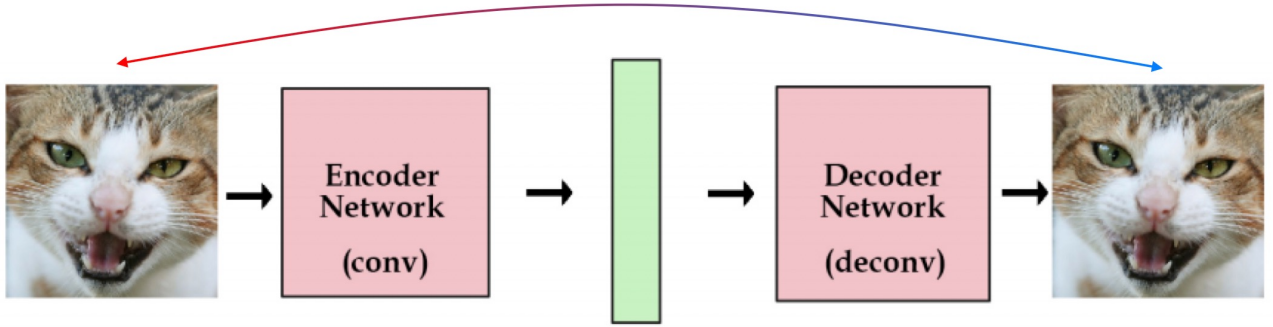
Encoder: Map from to a lower dimensional latent space

– Neural network $f_\theta(x)$ with parameters θ

Decoder: Map from latent space back to data space

– Neural network $g_\psi(z)$ with parameters ψ

Autoencoders



$$x \rightarrow z \rightarrow \hat{x}$$

$$x \rightarrow f(x) = z$$

$$z \rightarrow g(z) = \hat{x}$$

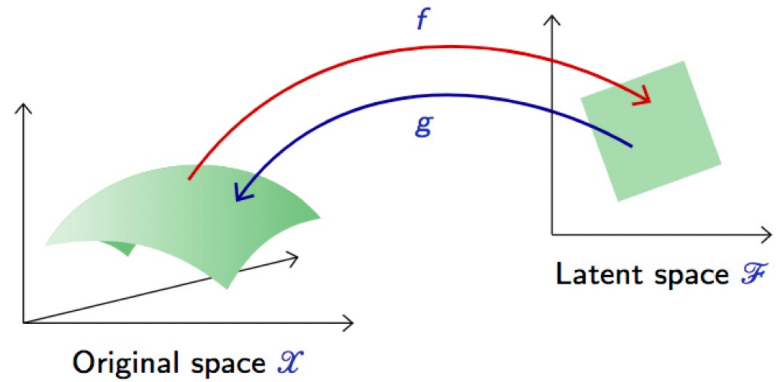
- Loss: Mean Reconstruction Error (MSE) between data and encoded–decoded data

$$L(\theta, \psi) = \frac{1}{N} \sum_n \left\| x_n - g_\psi(f_\theta(x_n)) \right\|^2$$

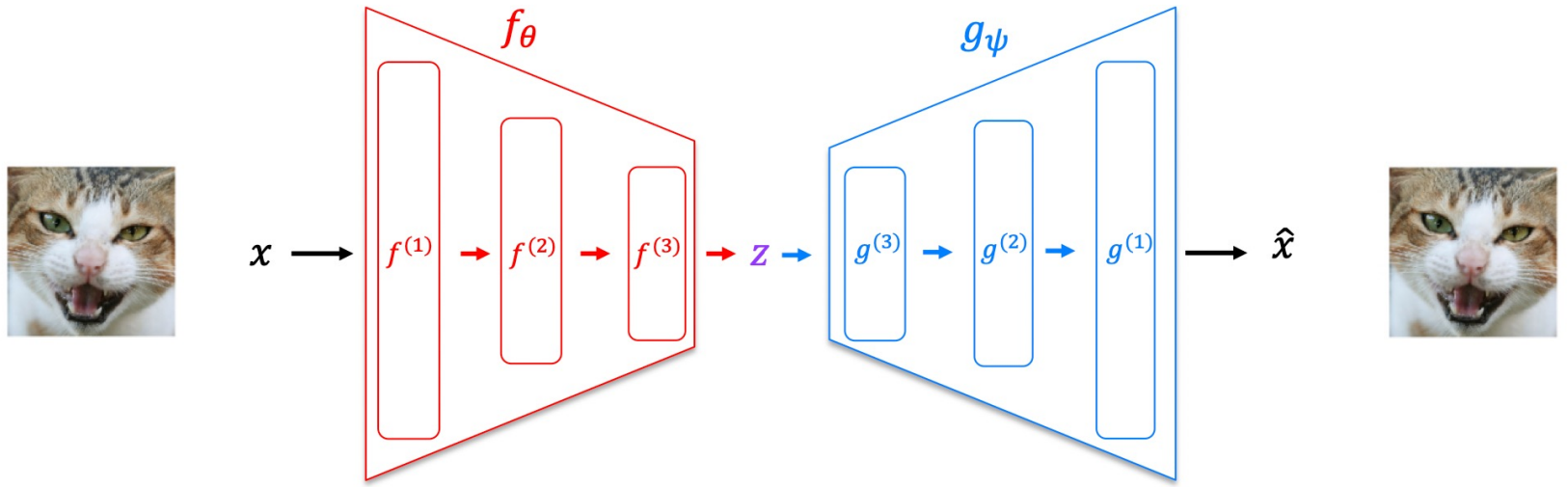
Minimize this loss over parameters of encoder (θ) and decoder (ψ).

We must: Choose latent dimension D, learn mapping functions $f(\cdot)$ and $g(\cdot)$.

Autoencoders



If the latent space is of lower dimension: autoencoder must capture a "good" parametrization, and in particular dependencies between components



- If f_θ and g_ψ are linear, optimal solution given by Principle Components Analysis (PCA). Autoencoders can be thought as a generalization of the PCA.
- When f_θ and g_ψ are multiple neural network layers, can learn complex mappings between x and z: f_θ and g_ψ can be Fully Connected, CNNs, RNNs, etc.

– Choice of network structure will depend on data

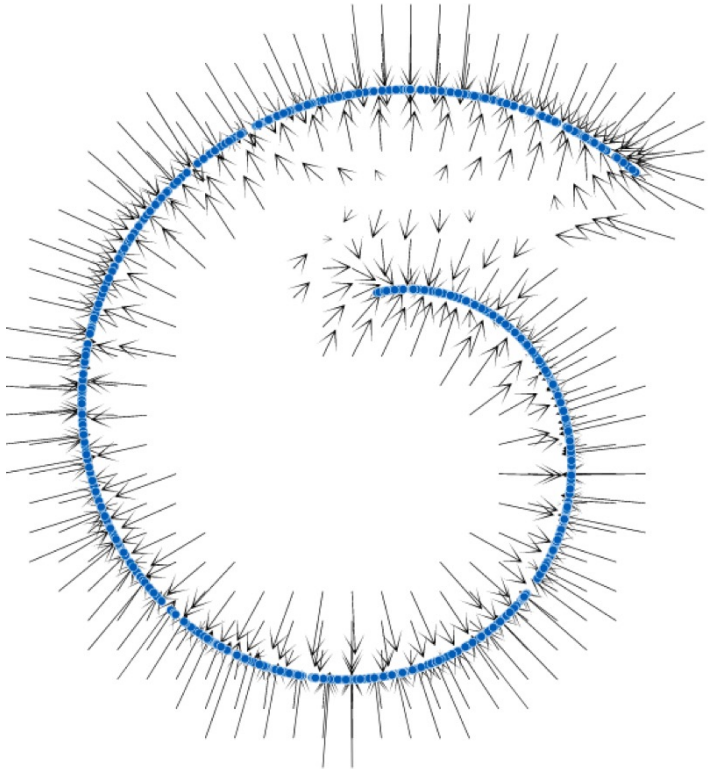
Autoencoders

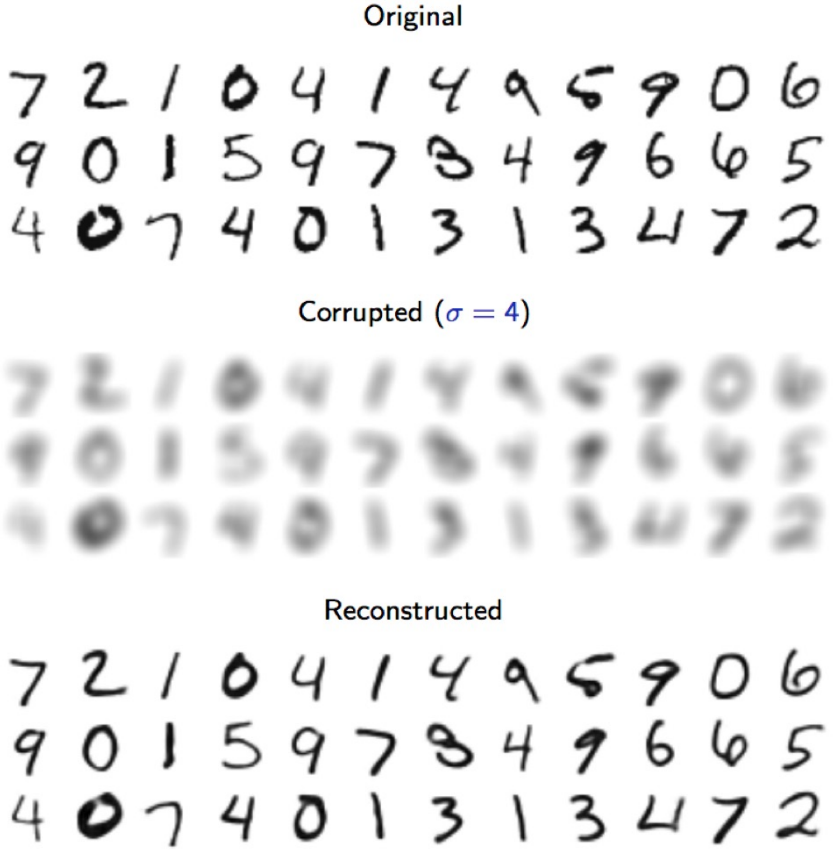
- Denoising Autoencoders: learn a mapping from corrupted data space $\tilde{\chi}$ back to original data space χ

– Mapping $\phi_w(\tilde{\chi}) = \chi$
– ϕ_w will be a neural network with parameters w

Loss:

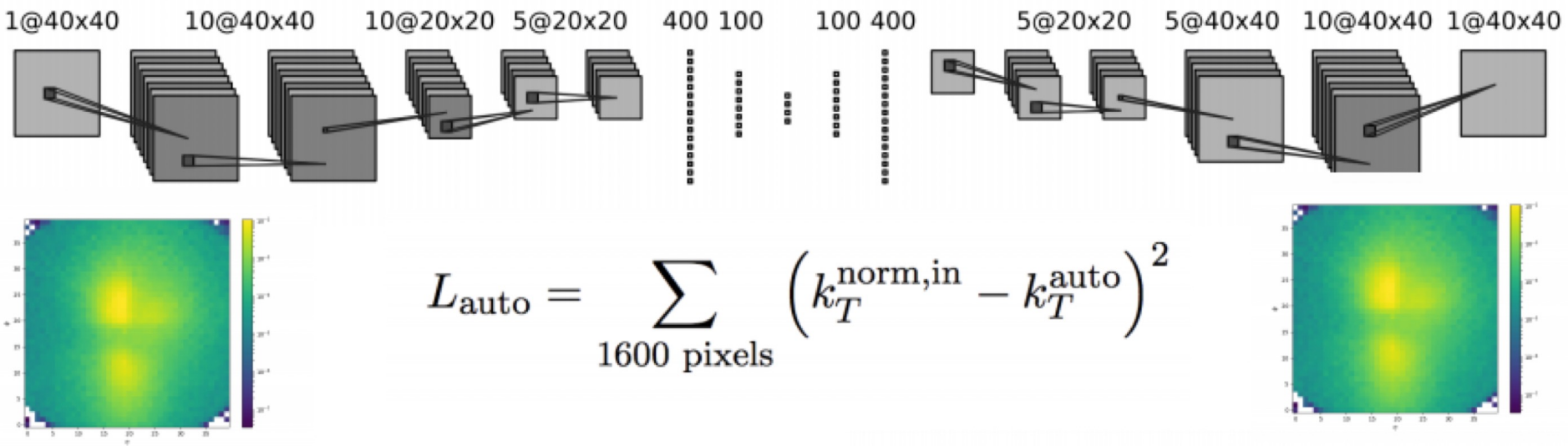
$$L(\mathrm{w}) = \frac{1}{N} \sum_n \|x_n - \phi_w(x_n + \epsilon_n)\|^2$$



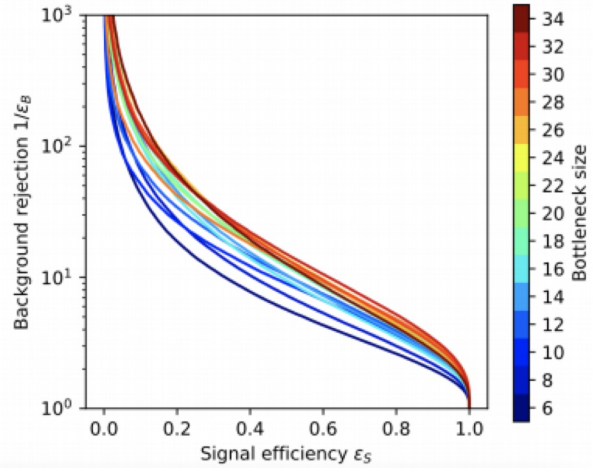


Autoencoders

- Anomaly detection: Find BSM (or top quark) jets in HEP training on pure QCD light quark and gluon jets and apply top tagging

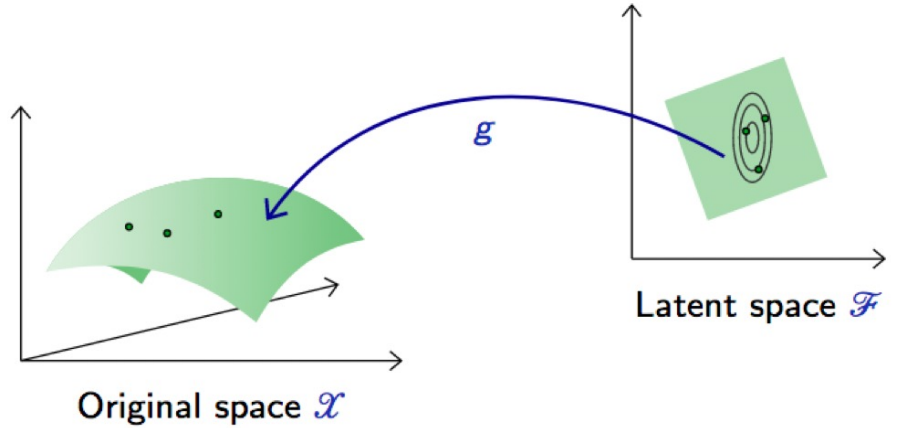


$$L_{\text{auto}} = \sum_{1600 \text{ pixels}} \left(k_T^{\text{norm,in}} - k_T^{\text{auto}} \right)^2$$



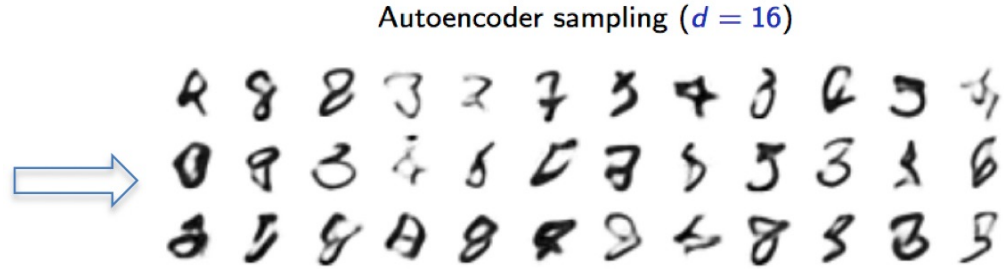
Decoders to simulate data

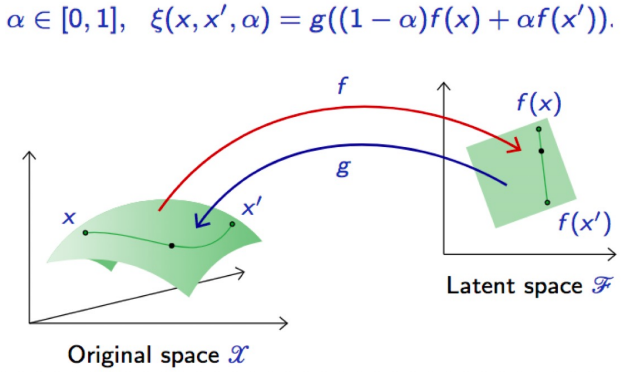
- Can we sample in latent space and decode to generate data?

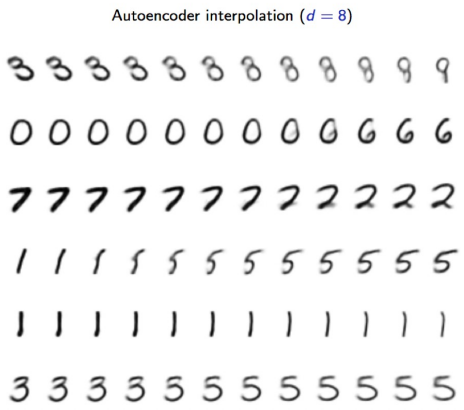


- What distribution to sample from in latent space?
  – Try Gaussian with mean and variance from data

Doesn't work! Don't know the right latent space density
– This can be done with a Variational Autoencoder







Parametrization trick
For $z \sim p_\theta(z)$, rewrite z as a function of a random variable ϵ whose distributions $p(\epsilon)$ does not depend on θ. Gaussian Example:
$z \sim \mathcal{N}(\mu,\sigma) \rightarrow \; z = \sigma * \epsilon + \mu \;\; where \; \epsilon \sim \mathcal{N}(0,1)$
VAE Loss:

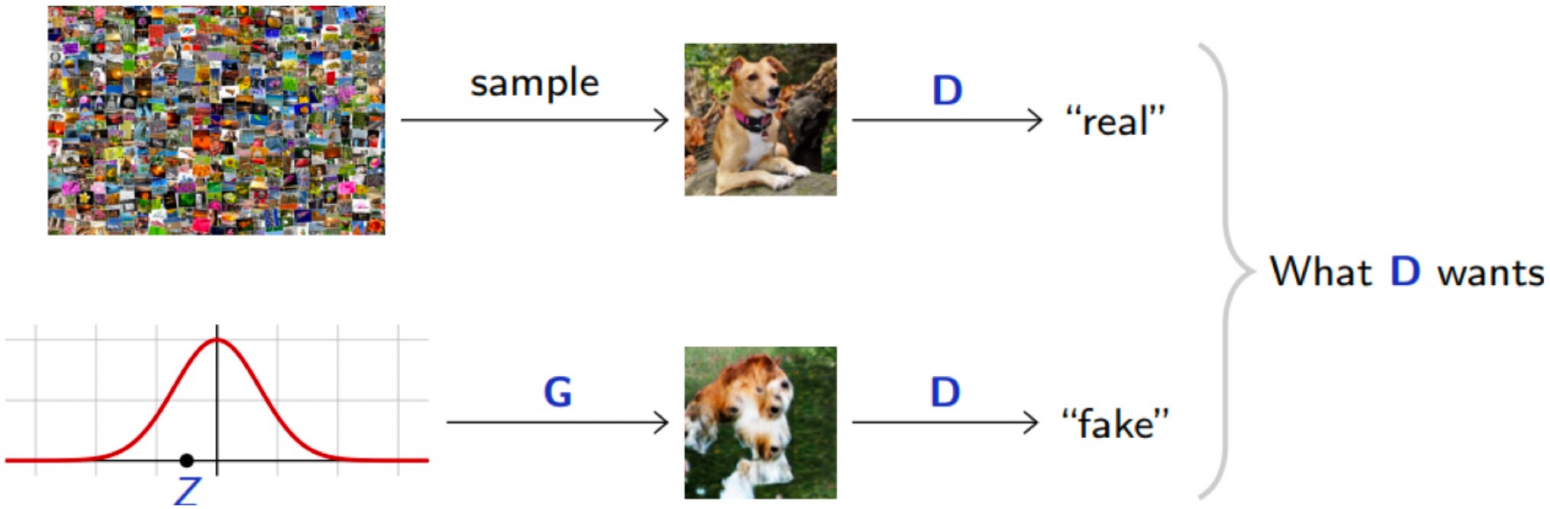
$$\max_{\theta,\psi} L(\theta,\psi) = \max_{\theta,\psi} \sum_{\epsilon \sim p(\epsilon)} \log p_\theta\left(x \middle| z_i = \epsilon * \sigma_\psi(x) + \mu_\psi(x)\right) - \log\left[\frac{q_\psi(z_i|x)}{p(z_i)}\right]$$

GAN

Generative Adversarial Network (GAN) models aim to:
Learn distribution p(x) that models pdf of the data and draw samples of plausible data points.
Explicit models if they can evaluate the p(x) or implicit models if they can only sample from p(x).

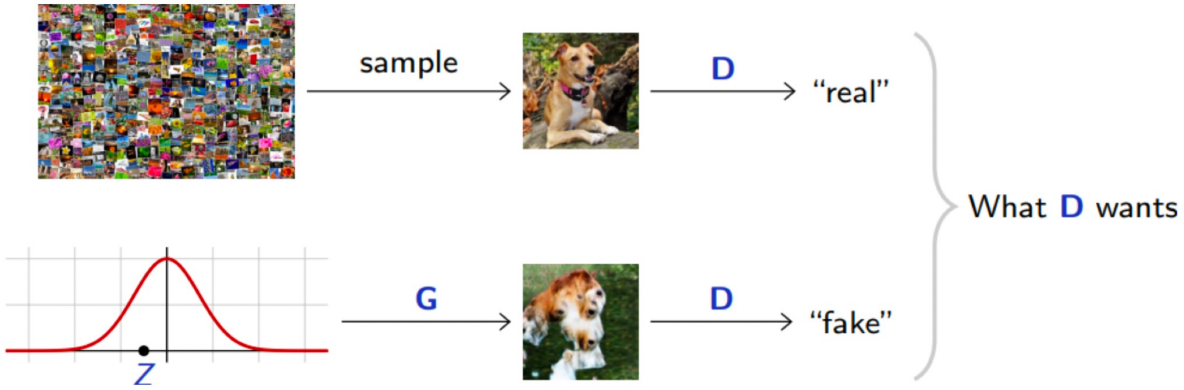
Two players game:

- Generator, network that is going to generate data that looks as less plausible as possible to real data
- Discriminator, network that is going to compare real data to fake data to determine which one is real or fake



- We need: a generator that can produce samples and a measure to estimate how a sample is far from being real or fake

GAN



- Generator network $g_\theta(z)$ with parameters θ

– Map sample from known $p(z)$ to sample in data space with z noise

$$x = g_\theta(z)$$

– We don't know what the learned distribution $p_\theta(x)$ is, but we can sample from Implicit Model

- Discriminator Network $d_\phi(x)$ with parameters ϕ

– Classifier trained to distinguish between real and fake data

– Classifier is learning to predict p($input = real\ x$)
– Classifier is our measure of not too far from the real data

- Generator goal:

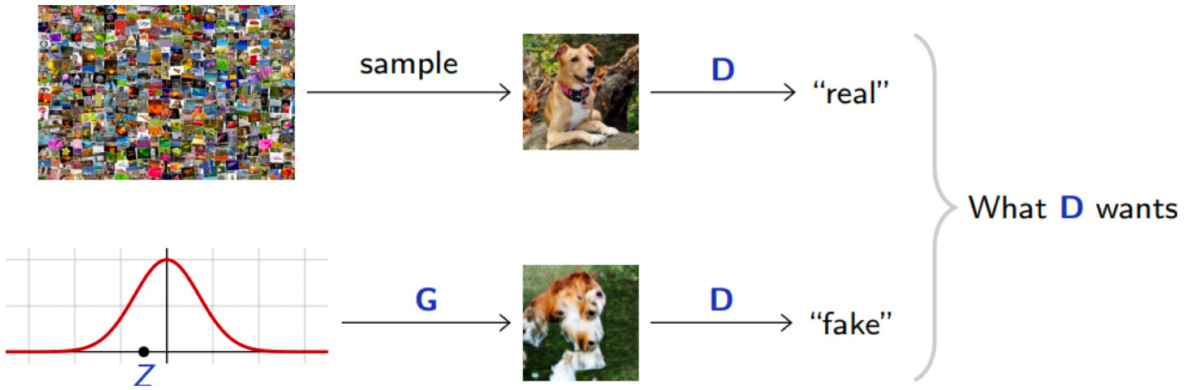
– Produce fake data to trick discriminator to classify as real

- Discriminator goal:

– Minimizes miss-classification of data as real or fake

- Adversarial setup: two networks with opposing objectives

GAN



Data

- Real data samples: $\{x_i, y_i = 1\}$
- Fake data samples: $\{\tilde{x}_i = g_\theta(z_i), \tilde{y}_i = 0\}$ with: $z_i \sim p(z)$

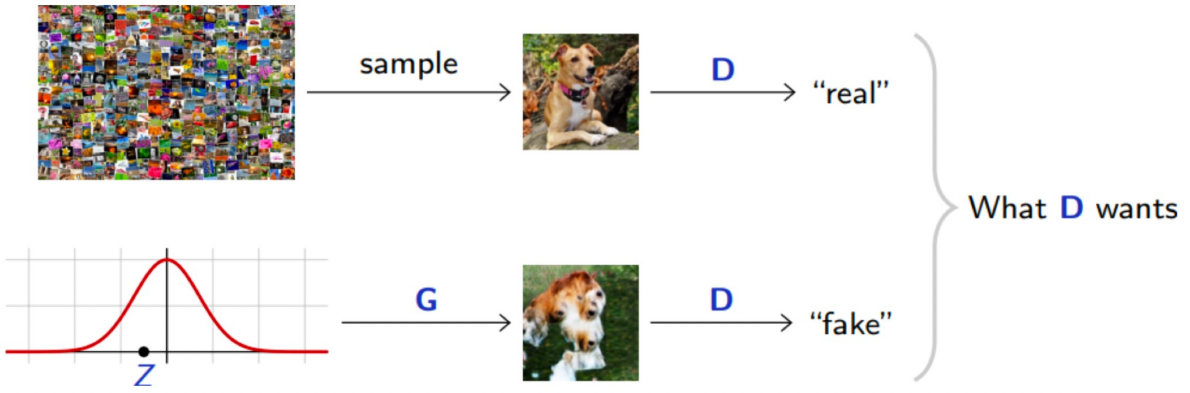
Usually Gaussian $\mathcal{N}(0,1)$

For a fixed generator, can train discriminator by minimizing the binary cross entropy

$$L(\phi) = -\frac{1}{2N}\sum_{i=1}^{N}\left[y_i \log d_\phi(x_i) + (1-\tilde{y}_i)\log(1-d_\phi(\tilde{x}_i))\right]$$

$$= -\mathbb{E}_{x\sim p_{data}(x)}\left[\log d_\phi(x_i)\right] - \mathbb{E}_{z\sim p(z)}\left[\log(1-d_\phi(g_\theta(z)))\right]$$

GAN



- Consider objective as a *value function* of ϕ and θ

$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log d_\phi(x) \right] + \mathbb{E}_{z \sim p(z)} \left[\log(1 - d_\phi(g_\theta(z)) \ ) \right]$$

– For fixed generator, $V(\phi, \theta)$ is high when discriminator is good, i.e. when generator is not producing good fakes
– For perfect discriminator, $V(\phi, \theta)$ is low when generator is good, i.e. when generator confuses discriminator

- So our optimization goal becomes:

$$\theta^* = \arg \min_\theta \max_\phi V(\phi, \theta)$$

NOTE: can prove that minimax solution corresponds to generator that perfectly reproduces data distribution

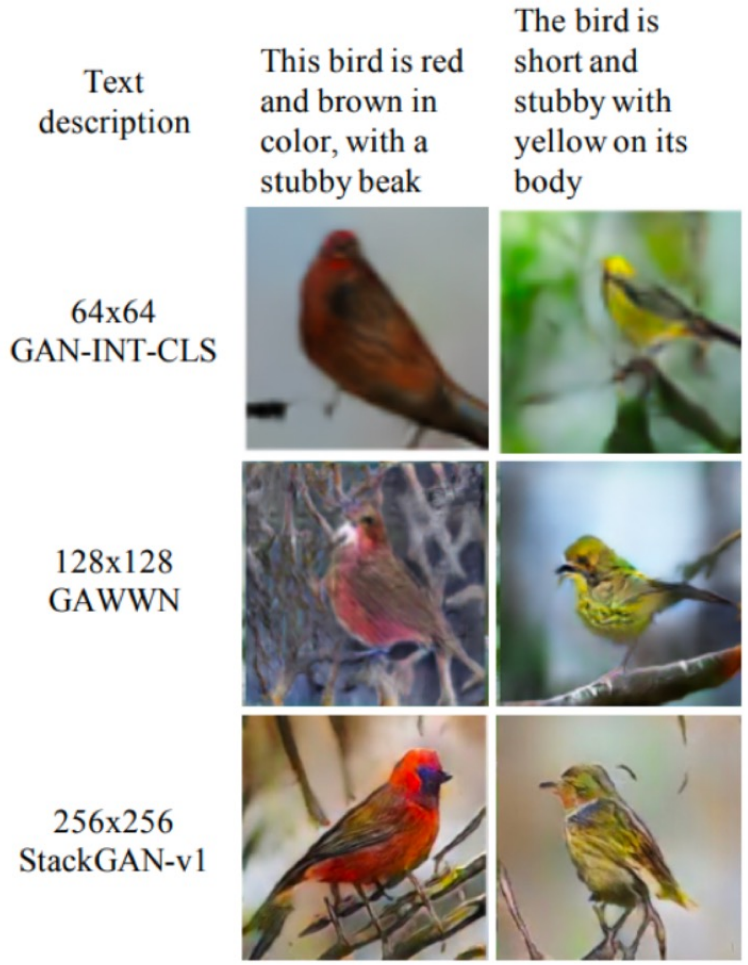
GAN

StyleGAN v2

(Karras et al, 2019)

Image-to-Image Translation with CycleGAN

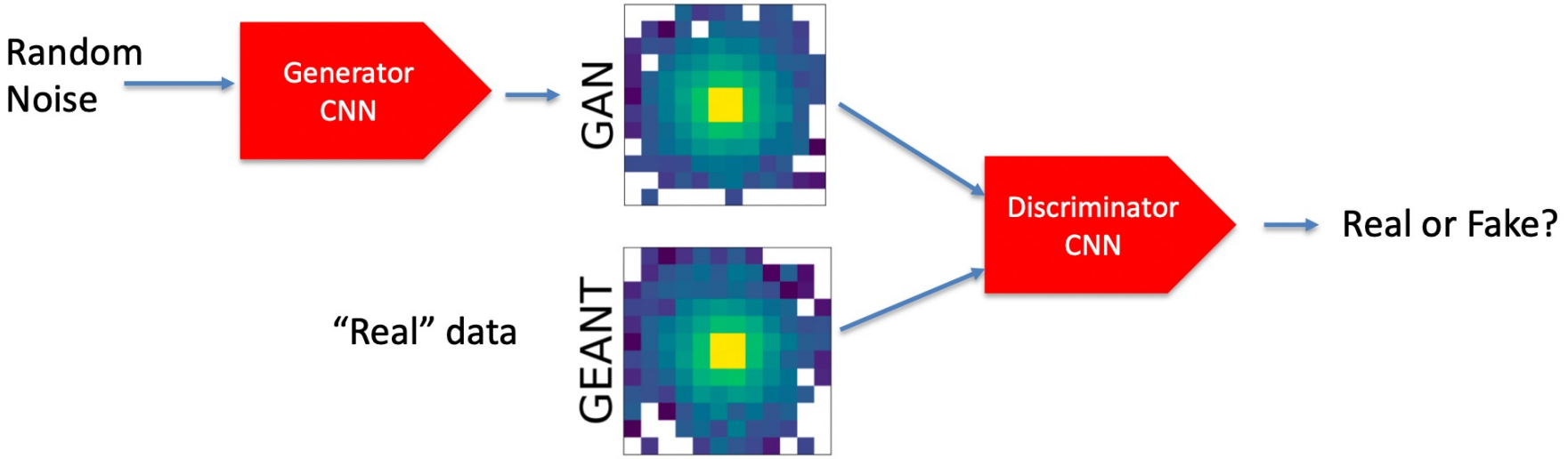
Text-to-Image Synthesis with StackGAN

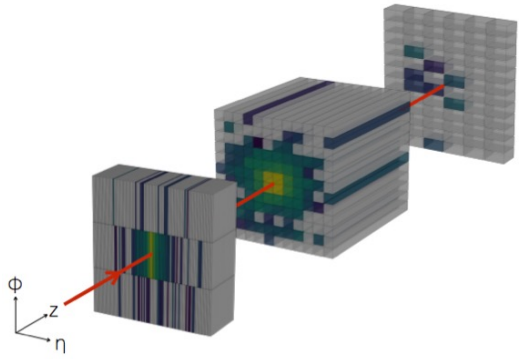


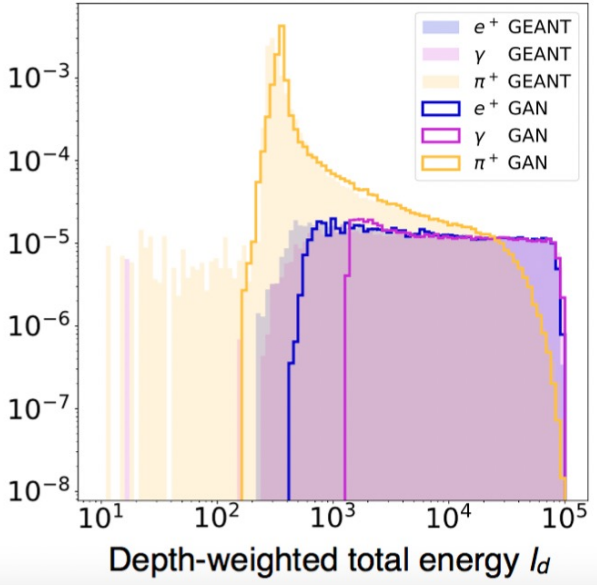
Zhang et. al. 2017

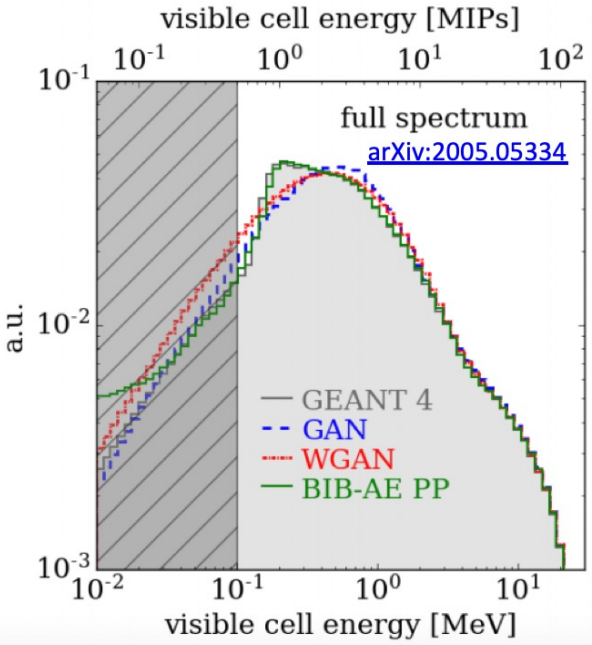
GAN

Calorimeter Energy Depositions with GAN







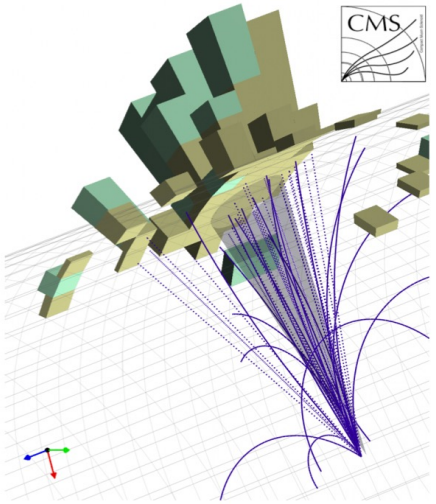


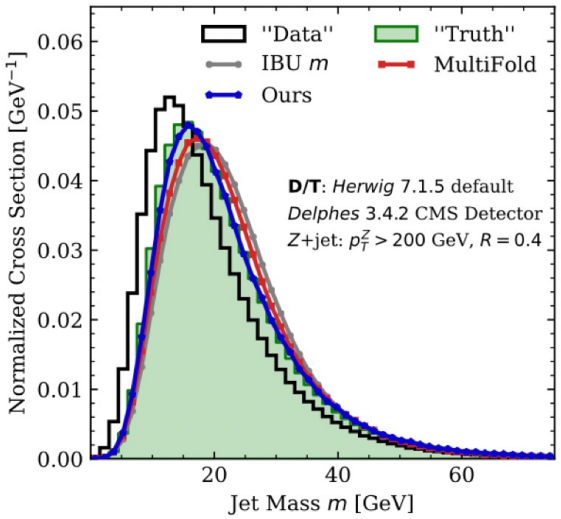
PRD97, 014021 (2018)
arXiv:1705.02355
arXiv:1701.05927

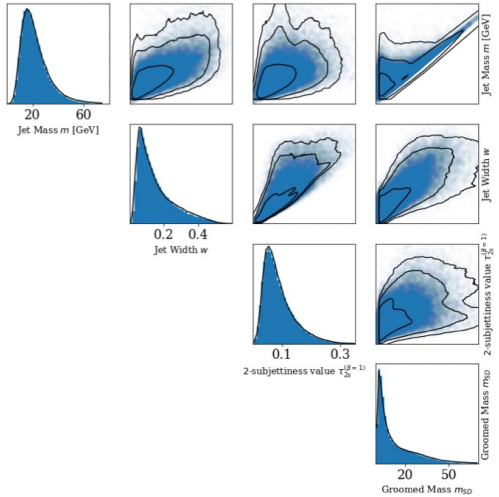
Conclusions

- Deep neural networks are an extremely powerful class of models
- We can express our inductive bias about a system in terms of model design, and can be adapted to a many types of data
- Even beyond classification and regression, deep neural networks allow for powerful model schemes such as Generative adversarial Networks that open many new possible tasks where Machine Learning can be applied in HEP.

Unfolding
Jet variables
in Z+jet events







PMLR 130:2107-2115, 2021

References

- M. A. Kagan, Machine Learning, CERN–Fermilab Summer School 2021
- Fleuret, Deep Learning Course, UNIGE
- Goodfellow et at, Generative Adversarial Nets

Other interesting lectures from CERN–Fermilab Summer School 2021

- S. Forte, Pertubative QCD & Jet Physics [1], [2], [3], UNIMI
- D. Bortoletto, Detector Technologies [2], Oxford
- S. Dawson, Higgs and Electroweak Theory [1], BNL
- R. Frederix, Monte Carlo Generators [1], [2], Lund