Simulation & Generative Models

Gregor Kasieczka Email: gregor.kasieczka@uni-hamburg.de Twitter/X: [@GregorKasieczka](https://twitter.com/GregorKasieczka) COFI Winter School 2023

CLUSTER OF EXCELLENCE

QUANTUM UNIVERSE

KEŠ

CENTER FOR DATA AND COMPUTING IN NATURAL SCIENCES

FSP

CMS

PUNCH DASHH *

Universität Hamburg and DESY

Partnership of

GEEÖRDERT VOM

Bundesministerium für Bilduna und Forschung

Motivation

Have: input examples (collision events, detector readouts, …)

Want: more data

Specifically: new data similar to the input, but not exact copies

How to encode in neural net?

Uses:

- Detector Simulation
- In-situ background estimation
- Surrogate models

• …

Overview

- 1. Common architectures*
	- -> GANs, VAEs, NF today
	- -> Diffusion & CNF tomorrow

*excluding transformers

ilianweng.github.io lilianweng.github.io 1406.2661 1406.2661

Training objective: Binary cross entropy

$$
\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log (1 - D(G(\mathbf{z})))]
$$
\n
$$
\uparrow
$$
\nTrue examples

\nTake examples

Training objective: Binary cross entropy

 $\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log (1 - D(G(\mathbf{z})))]$ \overline{G}

At (Nash) equilibrium: Generator produces realistic examples Discriminator is maximally confused

Training objective: Binary cross entropy

 $\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} [\log (1 - D(G(\boldsymbol{z})))]$

For generation: Sample from Generator Discard Discriminator

Comments on GANs

Architecture:

• Low complexity, fast and adaptable

Learning:

- Unstable training
- Matching of generator/discriminator (vanishing gradients)
- Mode collapse
- Loss function not interpretable

Maturity:

• Well established, many variants and extensions

Mode collapse

Wasserstein GAN

- Standard GANs minimise Jensen-Shannon divergence of generator output and true data
	- Not best measure, e.g. for non-overlapping distributions
- Replace with Wasserstein / Earth-Mover-Distance

$$
W_p(\mu,\nu)=\left(\inf_{\gamma\in\Gamma(\mu,\nu)}\mathbf{E}_{(x,y)\sim \gamma}d(x,y)^p\right)^{1/p}
$$

Wasserstein GAN

$$
\text{GAN loss:} \qquad \min_{G} \max_{D} \mathop{\mathbb{E}}_{\bm{x} \sim \mathbb{P}_r}[\log(D(\bm{x}))] + \mathop{\mathbb{E}}_{\tilde{\bm{x}} \sim \mathbb{P}_g}[\log(1 - D(\tilde{\bm{x}}))]
$$

Wasserstein GAN
\nloss^{*}: min max
$$
\underset{G}{\mathbb{E}} D\in \mathcal{D} \mathbf{x} \sim \mathbb{P}_r
$$
 $\left[D(\mathbf{x}) \right] - \underset{\tilde{\mathbf{x}} \sim \mathbb{P}_g}{\mathbb{E}} \left[D(\tilde{\mathbf{x}}) \right]$

Requires bounded Lipschitz norm, e.g. via term in loss

* Some mathematics involved from earth mover distance to here

Wasserstein GAN

$$
\text{GAN loss:} \qquad \min_{G} \max_{D} \mathop{\mathbb{E}}_{\bm{x} \sim \mathbb{P}_r} [\log(D(\bm{x}))] + \mathop{\mathbb{E}}_{\tilde{\bm{x}} \sim \mathbb{P}_g} [\log(1 - D(\tilde{\bm{x}}))]
$$

Wasserstein GAN $\min_{G}\max_{D\in\mathcal{D}}\mathop{\mathbb{E}}_{\bm{x}\sim \mathbb{P}_r}\left[D(\bm{x})\right]-\mathop{\mathbb{E}}_{\tilde{\bm{x}}\sim \mathbb{P}_q}\left[D(\tilde{\bm{x}})\right]$ loss:

> Improves training stability and sample quality (e.g. mode collapse)

Variational Autoencoders

Autoencoder

Two networks Encoder: data → latent space $\fbox{\parbox{1.5cm} \begin{tabular}{@{}c@{}} \hline $f(x)$ & $f(x)$ & \multicolumn{2}{|c@{}} \hline \multicolumn{2}{c}{f(x)} \hline \multicolumn{2}{c}{\textbf{Two networks}} \hline \multicolumn{2}{c}{\textbf{Encode:}}\ \textbf{data} \rightarrow \textbf{latent space} \rightarrow \textbf{data} \hline \end{tabular}}$

Autoencoder

Two networks Encoder: data \rightarrow latent space Decoder: latent space \rightarrow data

Training objective:

Autoencoder

Two networks Encoder: data \rightarrow latent space Decoder: latent space \rightarrow data

Training objective:
Minimise input/output difference $L = (x - f(g(x)))^2$ Decoder Encoder

Training objective:

Uses:

Dimension reduction **Denoising** Anomaly detection Generation?

Variational Autoencoder

 $f(x)=(\mu,\sigma)$

Variational Autoencoder (VAE): Split latent space

Variational Autoencoder

Variational Autoencoder (VAE): Split latent space Sample before decoder

$$
f(x) = (\mu, \sigma)
$$

$$
z = \text{Gaussian}(\mu, \sigma)
$$

$$
x'=g(z)
$$

Variational Autoencoder

Variational Autoencoder (VAE):

Split latent space Sample before decoder Penalty so mean/std are close to unit Gaussian

$$
f(x)=(\mu,\sigma)
$$

$$
z = \text{Gaussian}(\mu, \sigma)
$$

 $x' = g(z)$

$$
L = (x - g(z))^2 + \sigma^2 + \mu^2 - \log(\sigma) - 1
$$

(Calculate KL-divergence between Gaussians)

VAE Example

Latent space of MNIST VAE

towardsdatascience.com towardsdatascience.com

 $L = (x - g(z))^2 + \sigma^2 + \mu^2 - \log(\sigma) - 1$

How did we get here?

Sample from latent variables z $z_i \sim p(z)$

Produce data points x

$$
x_i \sim p(x \mid z)
$$

Sample from latent variables z $z_i \sim p(z)$

Produce data points x

$$
x_i \sim p(x \mid z)
$$

To choose correct latent distribution given data, could use Bayes theorem:

Conditional Prior
\n
$$
p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}
$$
 Difficult due to p(x)
\nEvidence

To choose correct latent distribution given data, could use Bayes theorem:

$$
p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}
$$

Instead, approximate with family of posterior distributions (variational inference):

$$
\mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x)) = \\ \mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)
$$

And find optimal approximation:

$$
q_\lambda^*(z \mid x) = \mathop{\arg\min}_{\lambda} \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))
$$

Still difficult due to (hidden) p(x) term!

And find optimal approximation:

$$
q_\lambda^*(z \mid x) = \mathop{\arg\min}_{\lambda} \mathbb{KL}(q_\lambda(z \mid x) \mid\mid p(z \mid x))
$$

Still difficult due to p(x) term!

jaan.io

$$
\mathbb{KL}(q_{\lambda}(z \mid x) \mid p(z \mid x)) =
$$

$$
\mathbf{E}_{q}[\log q_{\lambda}(z \mid x)] - \mathbf{E}_{q}[\log p(x, z)] + \log p(x)
$$

Introduce

$$
ELBO(\lambda) = \mathbf{E}_{q}[\log p(x, z)] - \mathbf{E}_{q}[\log q_{\lambda}(z \mid x)]
$$

Rewrite
$$
\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_{\lambda}(z \mid x) \mid p(z \mid x))
$$

As KL is >=0, ELBO is a lower limit for p(X) ELBO: Evidence Lower Bound

Maximise

$$
ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]
$$

Rewrite for samples, using neural networks:

$$
ELBO_i(\theta, \phi) = \mathbb{E}q_{\theta}(z \mid x_i)[\log p_{\phi}(x_i \mid z)] - \mathbb{KL}(q_{\theta}(z \mid x_i) \mid p(z))
$$
\n\nReconstruction term

\nAssume normal distribution

\n
$$
L = (x - g(z))^2 + \sigma^2 + \mu^2 - \log(\sigma) - 1
$$
Loss terms

Maximise

$$
ELBO(\lambda) = \mathbf{E}_{q}[\log p(x, z)] - \mathbf{E}_{q}[\log q_\lambda(z \mid x)]
$$

Rewrite for samples, using neural networks:

$$
ELBO_i(\theta, \phi) = \mathbb{E}q_{\theta}(z \mid x_i)[\log p_{\phi}(x_i \mid z)] - \mathbb{KL}(q_{\theta}(z \mid x_i) \mid p(z))
$$
\n\nReconstruction term

\nAssume normal distribution

\n
$$
L = (x - g(z))^2 + \sigma^2 + \mu^2 - \log(\sigma) - 1
$$

Comments on VAEs

Architecture:

- Low complexity, fast and adaptable
- Target: Maximise lower bound on likelihood

Learning:

- Stable training
- Average prediction → blurrier output
- Interpretable latent space

VAE

DCGAN

Maturity:

• Well established, many variants and extensions

Applications I

(Some) Simulation targets

Reduce computational bottleneck

Predict background from data

models developed for interesting contract contract contract contract contract contract contract contract contra Classification and generative modeling problems such as for fast emulation of GEANT4 calorimeter showers [9, 11]. Struction theirs **Reconstruction tasks**

Act as surrogate models

(Some) Simulation targets

Reduce computational bottleneck

Predict background from data

models developed for interesting contract contract contract contract contract contract contract contract contra Classification and generative modeling problems such as for fast emulation of GEANT4 calorimeter showers [9, 11]. Struction theirs **Reconstruction tasks**

Act as surrogate models

This happens in the experiment

This is what we want to know

Simulation is crucial to connect experimental data with theory predictions

This happens in the experiment

This is what we want to know

Simulation is crucial to connect experimental data with theory predictions, but computationally very costly years] ⋅

ATLAS Preliminary

2020 Computing Model -CPU: 2030: Baseline

This happens in the experiment

This is what we want to know

Simulation is crucial to connect experimental data with theory predictions, but computationally very costly

→Use generative models trained on simulation or data to augment simulations

Simulation targets

How to represent?

Simulation targets

How to represent?

Tabular data: Easy, insufficient for high-dimensions

Simulation targets

How to represent?

Tabular data

Fixed grid (voxels)

Generative results

Generative results

 z [layers]

 z [layers]

 z [layers]

 10^{-1} MeV

 z [layers]

Generative results

Go with the…

(Normalising) Flows

In auto-encoders, the decoder learns to 'undo' the encoder

Can we make this exact?

Learn a diffeomorphism between data and latent-space

Learn a diffeomorphism between data and latent-space

Bijective, invertable

Learn a diffeomorphism between data and latent-space

Bijective, invertable

Learn likelihood of data

Take into account Jacobian determinant to evaluate probability density

Invertible Easy-to-calculate Jacobean

Coupling flows

ule lilost expressive,
ion/understanding ו
ח x1
111. Coupling layers: Not the most expressive, but useful for illustration/understanding

Coupling flows

 \mathbf{S} Simple (e.g. dense) neural networks

Coupling flows

Easy-to-calculate Jacobian

260 *Deep Learning for Physics Research* **Calculating Jacobian determinant** the networks *sⁱ* and *tⁱ* themselves are not invertible — and do not need **the best contribution of the algebra** modern modern modern modern and always used in the overall block which are right halves of Figure 18.10 respectively — applying the following changes ta da

$$
\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \xrightarrow{f_1} \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{x}_2 \end{pmatrix} \xrightarrow{f_2} \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix} \text{ with } \begin{pmatrix} \mathbf{x}_1 \xrightarrow{f_1} \mathbf{z}_1 = \mathbf{x}_1 \odot \exp(s_2(\mathbf{x}_2)) + t_2(\mathbf{x}_2) \\ \mathbf{x}_2 \xrightarrow{f_1} \mathbf{x}_2. \end{pmatrix}
$$

$$
\mathbf{J}_{1} = \begin{pmatrix} \frac{\partial \mathbf{z}_{1}}{\partial \mathbf{x}_{1}} & \frac{\partial \mathbf{z}_{1}}{\partial \mathbf{x}_{2}} \\ \frac{\partial \mathbf{x}_{2}}{\partial \mathbf{x}_{1}} & \frac{\partial \mathbf{x}_{2}}{\partial \mathbf{x}_{2}} \end{pmatrix} = \begin{pmatrix} \text{diag}(\exp(s_{2}(\mathbf{x}_{2}))) & \frac{\partial \mathbf{z}_{1}}{\partial \mathbf{x}_{2}} \\ 0 & 1 \end{pmatrix}
$$

trangular by construction **Triangular by co.**
 Triangular by co. diag(exp(*s*2(x2))) @z¹ an
 $B = \frac{1}{2}$ and $B = \frac{1}{2}$ and $B = \frac{1}{2}$ and $B = \frac{1}{2}$ and $C = \frac{1}{2}$ and $C = \frac{1}{2}$ simplifies the calculation of the calculation of the determinant: the determinant: the determinant: the determinant:
The determinant: the determinant: the determinant: the determinant: the determinant: the determinant: the **Triangular by construction**

$$
\det \mathbf{J}_1 = \prod \exp(s_2(\mathbf{x}_2)) = \exp\left(\sum s_2(\mathbf{x}_2)\right)
$$

Composition

nneiti ltion c iomposition of bijective f x2 Figure 18.10: Example for an invertible mapping using a real-valued nonvolume preserving (real NVP) transformation [208]. Here, *sⁱ* and *tⁱ* (*i* = e filho λ $\mathbf{P}\mathbf{H}\mathbf{N}\mathbf{P}$ remains bijective **transformation and** *t* Composition of bijective functions

The *backward pass* operates with reversed signs. The division can be achieved by element-wise multiplication by exp (*si*) where *i* = 1*,* 2. While to be, as they are always used in forward mode — the overall block which mate between $\mathbf{C}(\mathbf{C})$ is interesting. For the invertible block to be useful in practice, we also need to calculate The *backward pass* operates with reversed signs. The division can be achieved by element-wise multiplication by exp (*si*) where *i* = 1*,* 2. While to be, as they are always used in forward mode — the overall block which maps between x and z is invertible. For the invertible block to be useful in practice, we also need to calculate The *backward pass* operates with reversed signs. The division can be Chain rule: Jacobian determinant of θ be, as they are always used in formar mode θ eren maar composition is product of determinants

How to train NF?

log likelihood of data Training objective: Minimise negative

||f(x))*||*² ². Also inserting the explicit form of the Jacobian determinant Udilipie po Sample points from training data

$$
\mathcal{L} = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[-\frac{1}{2} ||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x}) \right]
$$

How to train NF?

log likelihood of data Training objective: Minimise negative

How to train NF?

log likelihood of data Training objective: Minimise negative

$$
\mathcal{L} = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[-\frac{1}{2} ||f(\mathbf{x}))||_2^2 + \sum s(\mathbf{x}) \right]
$$

Contribution from Jacobian
determinant

$$
\det \mathbf{J} = \exp \left(\sum s(\mathbf{x}) \right)
$$

$$
- \log(\det \mathbf{J}) = -\sum s(\mathbf{x}) \quad \stackrel{\text{loc}}{\approx} \quad \stackrel{\text{loc}}{\approx}
$$

Animation

Autoregressive Flows relatively efficiently (Figure 4).

 $\overline{}$ Outputs conditioned on previous inputs Then an autoregressive model is a function ^g : **^R***^D* ! **^R***^D* $\mathbf{F}_{\mathbf{a}}$, and the left, is the direct autoregressive flows. On the dir Alternative to coupling flows: Alternative to coupling flows: the right, is the inverse autoregressive flow from Equation (20). Each

Autoregressive Flows relatively efficiently (Figure 4). flow and the propagate series for the part. The part of the part.

 Δ \mathbf{H} outputs \mathbf{H} Alternative to coupling flows: Alternative to coupling flows: Unce on provided inpute Alternative to coupling nows.
Outputs conditioned on previous inputs Δ Iternative to coupling flows[.]

where x1:*^t* = (*x*1*,...,xt*). For *t* = 2*,...,D* we choose flows in the "normalizing flow" direction (*i.e.*, in terms of f σ datuplian dirum mediudity in base density) to enable example σ Again: simple Jacobian and invertible functions. parameters, and ⇥¹ is a constant. The functions ⇥*t*(*·*) are

Autoregressive Flows relatively efficiently (Figure 4). flow and the propagate series for the part. The part of the part.

$$
y_t = h(x_t; \Theta_t(\mathbf{x}_{1:t-1}))
$$

Then an autoregressive model is a function ^g : **^R***^D* ! **^R***^D* **Masked** *y^t* = *h*(*xt*; ⇥*t*(x1:*t*¹))*,* (18) U Masked autoregressive flow (MAF):
 $space \rightarrow data$ Fast: Data → latent space
Slow: Latent space → data $\frac{1}{2}$ is a constant of the function space \rightarrow data Slow: Latent space → data

Autoregressive Flows relatively efficiently (Figure 4). **flow has higher a bijection and has higher part . All one one parts.** of the space and which is parameterized conditioned on

$$
y_t = h(x_t; \Theta_t(\mathbf{x}_{1:t-1})) \quad y_t = h(x_t; \theta_t(\mathbf{y}_{1:t-1}))
$$

$$
y_t = h(x_t; \Theta_t(\mathbf{x}_{1:t-1})) \quad y_t = h(x_t; \theta_t(\mathbf{y}_{1:t-1}))
$$

Then an autoregressive model is a function ^g : **^R***^D* ! **^R***^D* **Masked** *y^t* = *h*(*xt*; ⇥*t*(x1:*t*¹))*,* (18) outprographical depends on the current input and the contraction of acorputation is interested.
Computation is interested and cannot be parallelized. Masked autoregressive flow (MAF): Inverse autoregressive Fast: Data → latent space Slow: Data → latent
Slow: Latent space → data Fast: Latent space – Slow: Latent space → data Slow: Latent space → data **by a** *tast: Latent space –*

space → data Fast: Latent space → data low (IAF): of the log-likelihood during training. In this context one can be a set of the context one can be a set of this context one can be a set of the context one can be a set of the context one can be a set of the context one ca think of IAF as a flow in the generative direction: *i.e.*in terms essive flow (MAF): Inverse autoregressive flow (IAF): Slow: Data → latent space α is the same as the same as the same as the same as the same of the inverse of the the fact: Early in Equation (18), hence the name. Computation of the nam

Comments on Flows

Only scratched the surface: more constructions available

Comments on Flows

Only scratched the surface: more constructions available

Exact learning of likelihood

- \rightarrow Better generative fidelity
- \rightarrow Can evaluate likelihood of data

More complex

→ Slower, choice of fast direction

Simulation & Generative Models

Redit

Gregor Kasieczka Email: gregor.kasieczka@uni-hamburg.de Twitter/X: [@GregorKasieczka](https://twitter.com/GregorKasieczka) COFI Winter School 2023

CLUSTER OF EXCELLENCE

QUANTUM UNIVERSE

CDCS

KEŠ

CENTER FOR DATA AND COMPUTING IN NATURAL SCIENCES

FSP

CMS

PUNCH DASHH *

Partnership of Universität Hamburg and DESY GEEÖRDERT VOM

Bundesministerium für Bilduna und Forschung

Yesterday

First look at simulating fixed grid data

Overview

1. Common

architectures

- -> GANs, VAEs, NF yesterday
- -> Diffusion & CNF today

This paper presents probabilities in diffusion probabilities in the sense of the probability of the sense (which we will call a "diffusion" for brevitsing model will call a parameterized Markov chain trained using using using the set of the Core idea: Stepwise transition from pure noise to data

Warkov chain inference to produce samples matching the data after finite time. This can be a finite that a finite time. This can be a finite are learned to reverse a diffusion process, which is a Markov chain that gradually additionally additionally a
That gradually additionally additionally additionally additionally additionally additionally additionally addi

\mathbf{Di} ordering that vastly generalizes what is normally possible with autoregressive models. x1*,...,* x*^T* are latents of the same dimensionality as the data x⁰ ⇠ *q*(x0). The joint distribution *p* if is a process of the *repeat of the process*, and it is defined as a matrix α m

This is added in the training proposed (Section 4.2), so we consider the one of our primary consider the one of our primary contributions. The one of our primary contributions of our primary contributions. The contributions of our primary contributions process **both process** *^t* are small [53]. A notable property of the forward process is that it admits sampling x*^t* at an process

Diffusion Model Diffusion Products x1*,...,* x*^T* are latents of the same dimensionality as the data x⁰ ⇠ *q*(x0). The joint distribution *^q*(x1:*^T [|]*x0) := ^Y *q*(x*t|*x*^t*¹)*, q*(x*t|*x*^t*¹) := *N* (x*t*; posterior *q*(x1:*^T |*x0), called the *forward process* or *diffusion process*, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule 1*,..., ^T* :

Shortcut:
transition to

This is added in the training the sample our best sample of the same of the training proposed (Section 4.2), so we consider the one of our primary consider the one of our primary contributions. The one of our primary contributions of our primary contributions. The contributions of our primary contributions This is added in the training α_t . If β process **both process** *^t* are small [53]. A notable property of the forward process is that it admits sampling x*^t* at an process *q*(x*t|*x0) = *N* (x*t*; $\frac{1}{2}$ $\frac{1}{2}$

Diffusion Model x1*,...,* x*^T* are latents of the same dimensionality as the data x⁰ ⇠ *q*(x0). The joint distribution *^q*(x1:*^T [|]*x0) := ^Y *q*(x*t|*x*^t*¹)*, q*(x*t|*x*^t*¹) := *N* (x*t*;

proposed (Section 4.2), so we consider the one of our primary consider the one of our primary contributions. The one of our primary contributions of our primary contributions. The contributions of our primary contributions process **both process** *^t* are small [53]. A notable property of the forward process is that it admits sampling x*^t* at an process 2² .
hi hi rd S IS
*D*Ces x*t*(x0*,* ✏)*,*

Diffusion Model phenomenon in the language of lossy compression, and we show that the sampling procedure of \mathbf{S} interview is a sample our best sample quality results using the sample quality results using the sample \mathbf{S}

Optimal expected mean (take into account known noise schedule)
account known noise schedule) *q*(x*t|*x*^t*¹)*, q*(x*t|*x*^t*¹) := *N* (x*t*;

parameterization (Section 4.2), so we consider this equivalence to be one of our primary contributions.

Reverse Resultsing learning objective (Noise
$$
\rightarrow
$$
 data)

\n
$$
L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\left\| \boldsymbol{\epsilon} - \frac{\boldsymbol{\epsilon}_{\theta}}{\boldsymbol{\epsilon}} (\sqrt{\bar{\alpha}}_t \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}}_t \boldsymbol{\epsilon}, t) \right\|^2 \right]
$$
\nNoisy image

\nReminder: Forward diffusion to time t

\n
$$
\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}}_t \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}}_t \boldsymbol{\epsilon}
$$
\nTimes the function of the system is given by:

parameterization Γ imestep Γ (*L^T* does not appear because the forward process variances *^t* are fixed.) Algorithm 1 displays the Timestep

from pure noise to data (which we will call a "diffusion model" for brevity) is a parameterized Markov chain trained using Core idea: Stepwise transition

data in the opposite direction of sampling until signal is destroyed. When the diffusion consists of Algorithm 1 Training chain to set the sampling chain transition of Gaussian transitions to conditions to condi

- d: repeat 1: repeat
- 2. $\mathbf{X}_0 \sim q(\mathbf{X}_0)$
2. $\mathbf{X}_0 \sim q(\mathbf{X}_0)$ 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
- 3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\mathbf{s} \in \mathbb{R}^d$ is a subset of generation of generation models are capable of \mathbf{s} . 4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient 5: Take gradient descent step on

$$
\nabla_{\theta} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta} (\sqrt{\overline{\alpha}_{t}} \mathbf{x}_{0} + \sqrt{1 - \overline{\alpha}_{t}} \boldsymbol{\epsilon}, t) \right\|^{2}
$$

6: until converged

s. until converged multiple noise levels during training training and with an annualed Langevin dynamics and with an 6: until converged

from pure noise to data (which we will call a "diffusion model" for brevity) is a parameterized Markov chain trained using Core idea: Stepwise transition

Core idea: Stepwise transition

 \mathcal{L} https://medium.com/mlearning-ai/enerating-images-withddpms-a-pytorch-implementation-cef5a2ba8cb1

from pure noise to data Core idea: Stepwise transition

CaloCloud, time stamp: t₉₉

from pure noise to data (which we will call a "diffusion model" for brevity) is a parameterized Markov chain trained using Core idea: Stepwise transition

 $DALLE-2$ $(Roughly)$ $\frac{1}{10}$ $\frac{1}{5}$ *Contrastive learning of joint image/text embedding + translation of embedding +* **conditioned diffusion** $s_{\text{in}} = s_{\text{in}} = s_{\text{in}} = s_{\text{out}}$ (i.e. s_{out}), some s_{out} and s_{out} are s_{out} samples, some s_{out} better than the published results of generative models of generative models ($\frac{1}{2}$ generation $\frac{1}{2}$ $\frac{\partial S}{\partial \alpha}$

https://openai.com/dall-e-3; https:// www.assemblyai.com/blog/how-dalle-2-actually-works/ $\frac{p}{p}$ \geq

Continuos Diffusion Model

external over models, where the basic internal differential equations with stochastic differential equations α equation (SDE). To reverse the SDE and generate samples, the score function is needed. Figure (SDEs)with deconduction annorontial oquation 3.2) parameterization (Section 4.2), so we consider this equivalence to be one of our primary contributions. with stochastic differential equations

Continuos Di such that xp0q " *p*0, for which we have a dataset of i.i.d. samples, and xp*T*q " *p^T* , for which we **diagree a tractable form to generate summarize samples effects** effects and an experimental contribution and distribution and distribution and distribution and distribution and distribution \boldsymbol{p}

Forward SDE:

(Correspond to the noise schedule in discrete case)

Continuos Diffusion Model A remarkable result from Anderson (1982) states that the reverse of a diffusion process is also a

diffusion processing processing backwards in the reverse-time support $\mathsf{X}(\mathsf{U})$ Probability density of x(t)

Reverse SDE:
$$
dx = [f(x, t) - g(t)^2 \nabla_x \log p_t(x)]dt + g(t) d\bar{w}
$$

\nScore function

Reverse of a diffusion process is also a diffusion **procession** and **property** it to sample from *process* is also a diffusion

Continuos Diffusion Model \sim remarkable result from Anderson (1982) states that the reverse of a diffusion process is also a diffusio for all *t*, we can derive the reverse diffusion process from Eq. (6) and simulate it to sample from *p*0.

$$
\text{Reverse SDE:} \quad \mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x},t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t) \mathrm{d}\bar{\mathbf{w}}
$$

$$
\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t) | \mathbf{x}(0)} \Big[\left\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \right\|_2^2 \Big] \Big\}
$$

Learn to approximate score function with neural network

Continuos Diffusion Model

High Fidelity Particle Cloud Generation with Flow Matching Cedric Ewen Once trained: Sample latent space and numerically solve SDE to transport to data space

Aside

Well, almost

Flow Matching

Remember: Normalising Flows " ||
|
| *i*
∂filici∏ −1 −1.11.000 *,* (4.6)

$$
\log p_0(x_0) = \log p_T(x_T) - \sum_t \log |\det J_t|
$$

Remember: Normalising Flows

Apploximate vector field with network Approximate vector field with network

Remember: Normalising Flows In order to express complex densities it makes sense to use a sequence of invertible functions *f^t*

Remember: Normalising Flows

 $x_0 = f_1^{\theta_1} ... \circ f_{T-1}^{\theta_{T-1}} \circ f_T^{\theta_T}(x_T)$

discrete normalizing flow discrete normalizing flow $\frac{dx_t}{dt} = v_t^{\theta}(x_t), \quad x_t := f(x_T, t)$

For sampling: The sampling flow to a continuous flow t Solve differential equation (ODE) pendent invertible functions *f^t* that are applied sequentially. By assuming that the independent

Remember: Normalising Flows

$$
x_0 = f_1^{\theta_1} \dots \circ f_{T-1}^{\theta_{T-1}} \circ f_T^{\theta_T}(x_T)
$$

discrete normalizing flow discrete normalizing flow

$$
\frac{dx_t}{dt} = v_t^{\theta}(x_t), \quad x_t := f(x_T, t)
$$

 F_1 ulitivity vooriy. $\frac{1}{2}$ out $\frac{1}{2}$ is a consistent flow consistent flow consistent in the constant of independent of independent independent independent independent independent independent independent in the constant independent ind *Calculate frace for criarige in probability volume* Training costly: Solve ODE to evaluate p(x) Calculate trace for change in probability volume

Flow Matching

 $\frac{dx_t}{dt} = v_t^{\theta}(x_t), \quad x_t := f(x_T, t)$

Training costly: Solve ODE to evaluate p(x) Calculate trace for change in probability volume Instead, learn to approximate target vector field F_1 ulitivity vooriy. $\frac{1}{2}$ out $\frac{1}{2}$ is a consistent flow consistent flow consistent in the constant of independent of independent independent independent independent independent independent independent in the constant independent ind pendent invertible functions *face* for change in probability. Volume

Flow Matching Flow Matching Center *^pt*(*x*) = ! *dx*⁰ *pt*(*x|x*0)*p*0(*x*0)*.* (4.15)

data sample *x*0 at α *t* α How to get the vector field **path which is a position of the data distribution** \overline{a} How to get the vector field describing morphing?

> Constrains: At t=0: Input data At t=1: Target latent distribution

Use mixture of conditional morphings

the marginal probability path which is a good approximation of the data distribution *p*⁰ at *t* = 0: *^pt*(*x*) = ! *dx*⁰ *pt*(*x|x*0)*p*0(*x*0)*.* (4.15) *^pt*(*x*) = ! *dx*⁰ *pt*(*x|x*0)*p*0(*x*0)*.* (4.15) The marginal vector field for generating this probability path can also be calculated by marginalizing over the conditioned vector fields *ut*(*x|x*0) and is given by *dx*⁰ *ut*(*x|x*0) *pt*(*x|x*0)*p*0(*x*0) *^pt*(*x*) *.* (4.16) The calculation of the marginal probability path and the marginal vector field is still not trivial Marginal path conditional path

Use mixture of conditional
\nmorphings

\n
$$
u_t(x) = \int dx_0 \, u_t(x|x_0) \frac{p_t(x|x_0)p_0(x_0)}{p_t(x)}
$$
\nWould allow calculating corresponding
\nvector field

 $\mathcal{L}_{\text{FM}} = ||v_{\theta}(x_t|t) - u_t(x_t|x_0)||^2$ $\begin{array}{ccc} \mathsf{Still} \end{array}$ expensive, let network only $\frac{1}{2}$ approximate conditional vector field $\mathcal{C}_{\text{EM}} = ||y_0(r, |t) - y_1(r, |r_0|)^2$ $\mathcal{L}_{FM} = ||v_{\theta}(x_t|t) - u_t(x_t|x_0)||^2$ approximate conditional vector field

Flow Matching 4.5.3 Optimal Transport Gaussian Conditional Probability Paths

How to get the vector field describing morphing?

and vector fields. A natural probability paths are conditioned as a natural probability paths and vector field was used to allow used to allow used to allow used that G is than unit G the right plot, the loss \mathcal{F} objective from Sec. 4.5.5 additionally enforces optimal transport paths between the marginal transport paths be a specific choice was a simple linear interpolation of the simple linear interpolation: α

Constrains: At t=0: Input data At t=1: Target latent distribution α (xlx) = Λ (xla) σ) choice is to use Gaussian probability paths of the form of the for 4.5.3 Optimal Transport Gaussian Conditional Probability Paths

Use mixture of conditional morphings

But what are the actual paths? Can use Gaussians with linear interpolation \bm{u} tual paths? $\frac{1}{2}$ to the unit Gaussian distribution and $\frac{1}{2}$ ($\frac{1}{2}$, $\frac{1}{2}$ s and s with a hyperparameter that controls the minimum standard deviation of the Gaussian of the Gaussia ${\rm s?}$ using this choice, the vector field from Eq. 4.20 and the loss function become ${\rm s?}$

aterit distribution $p_t(x|x_0) = \mathcal{N}(x|\gamma_t, \sigma_t)$ ${\sf orphings}$ and $\alpha^{\rm FM} = (1-t)x_0$ if what are the interestion on the initial state of $\sigma_t^{\text{FM}} = \sigma_{\min} + (1 - \sigma_{\min})t$ α specific choice was a simple linear interpolation: $\gamma_t^{\rm FM} = (1-t)x_0,$ $\sigma_t^{\text{FM}} = \sigma_{\text{min}} + (1-\sigma_{\text{min}}) t,$ where σmin is a hyperparameter that controls the minimum standard deviation of the Gaussian ρ or conquisition are considered to paths. This can be optimal in the sense that are considered to be optimal in the sense that ρ at are the $\sigma_t^{\text{FM}} = \sigma_{\min} + (1 - \sigma_{\min})t$ $\begin{array}{cc} \textbf{1} & \textbf{0} & \textbf{0} & \textbf{0} \\ \textbf{0} & \textbf{0} & \textbf{0} & \textbf{0} \\ \textbf{0} & \textbf{0} & \textbf{0} & \textbf{0} \end{array}$ $\frac{1}{2}$ are the shortest and straightest paths between the initial and final distribution. Straightest paths between the initial and final distribution. Since $\frac{1}{2}$

Vector field:

Loss:

this probability path but one transformation that leads to a simple vector field is the flow probability paths. This choice leads to paths that are considered to be optimal in the sense that *ut*(*xt|x*0)= ˙γ*^t* + ˙σ*t*# = (1 − σmin)# − *x*⁰ (4.23) *ut*(*xt|x*0)= ˙γ*^t* + ˙σ*t*# = (1 − σmin)# − *x*⁰ (4.23)

$$
\text{Loss:} \qquad \mathcal{L}_{\text{FM}} = \left\| (v_{\theta}(x_t, t) - (1 - \sigma_{\min})\epsilon - x_0) \right\|^2 \quad \underset{\text{N}}{\overset{\text{N}}{\text{O}}} \quad \underset{\text{N}}{\overset{\text{N}}{\text{O}}} \quad \text{S}
$$

2210.02747 $\frac{1}{\sqrt{2\pi}}$ $\frac{1}{2}$ $\tilde{\mathcal{C}}$

High Fidelity Particle Cloud Generation with Flow Matching Cedric Ewen **Optimal Transport**

Figure 4.4: Illustration of the condition of the condition of the flow matching framework. The left of the left o paths by: Can further improve

deviation around *x*⁰ to the unit Gaussian distribution at*t* = 1. Both other plots show an extension **Conditioning on**

start- and endpoint intransport condition Adding optimal transport condition

Optimal Transport

Conditioning on Adding optimal start- and endpoint

Adding optimal transport condition

Comments

equation (SDE). To reverse the SDE and generate samples, the score function is needed. Figure Score matching objective Flow matching objective

Flow matching objective plot shows the probability paths from Ref. [75] that go from a Gaussian with a small standard

deviation around *x*⁰ to the unit Gaussian distribution at*t* = 1. Both other plots show an extension of this framework introduced introduced introduced in Ref. [187]. In the middle plot, not all paths converge to a unit Gaussian which allows a transformation from any arbitrary distribution into another one. In respectively. By reversing this process, points drawn from the prior can be transformed into samples that follow the distribution. This reverse process is also described by a show equivalence for Gaussian probability paths

Continuous Normalising Flow introduced in the next section covering the training behaviour and especially the relation of more general framework (other T and α solver with any standard α solver when the sol definition of paths)
Comments

Close relation of CNF and diffusion models

Maturity: Very recent, much in flux

> Sample quality: Very high

Training: Stable $frac{1}{\text{Train}}$

Sampling: 2006. articles Diagonal Probability: 2020. articles Expensive Modeling through Stockastic Diagnosis of the Stockhastic Diagnosis Diagnosis Company of the Stockhastic Diagnosis Diagnosis Company of the Stockhastic Diagnosis Diagnosis Diagnosis Diagnosis Diagnosis Diagnosis D (however: consistency distillation) 12Yang Song et al. *Consistency Models*. 2023. arXiv: 2303.01469

Applications II

Comments on Flows

Exact learning of likelihood

 \rightarrow Better generative fidelity

 \rightarrow Can evaluate likelihood of data

More complex

 \rightarrow Slower, choice of fast direction

Generative results II Meliei anne legalite il

How to flows for high-dimensional data?

Generative results II

Simulation targets

How to represent?

Tabular data

Fixed grid (voxels) Limiting for high-dimensions (sparse data)

Point clouds / graphs

Simulation targets

Before tackling showers in calorimeters: Look at jet constituents (JetNet data): 3 features per constituents up to 30/150 constituents/jet

How to represent?

Tabular data

Fixed grid (voxels) Limiting for high-dimensions (sparse data)

Point clouds / graphs

Why? Useful stepping stone In-situ background

Point Clouds

- Example: *Sensors in a space*
	- Fixed grid vs arbitrary positions
	- Potential sparsity of data
- Permutation symmetry
- Can view as trivial graph

Total data
$$
\{\times\}_{j=1...}^{r}
$$
 $\overline{P_{xamples}}\}$
\n $w_{i}th \times V = \{\overline{P_{1}^{y}}, \overline{P_{1}^{y}}, \overline{P_{L(j)}^{y}}\}$
\nand $\overline{P_{i}} \in \mathbb{R}^{D}$

Deep Sets

Theorem 7 Let $f : [0,1]^M \to \mathbb{R}$ be a permutation invariant continuous function iff it has the representation \mathbf{r}

$$
f(x_1, ..., x_M) = \rho \left(\sum_{m=1}^{M} \phi(x_m) \right)
$$
 (18)

for some continuous outer and inner function $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ respectively. The inner function ϕ is independent of the function f.

$$
x = \frac{p_1}{p_2} = [r_1, q_1, q_1, T_1]
$$
\n
$$
= [r_1, q_2, q_1, T_2]
$$
\n
$$
f(x) = g(\sum_{i=1}^{n} \phi(\overrightarrow{p_i}))
$$
\n
$$
f(\overrightarrow{p_i}) = \sum_{i=1}^{n} \phi(\overrightarrow{p_i})
$$
\n
$$
f(\overrightarrow{p_i}) = \sum_{
$$

How to GAN with it

 $r \alpha$

Generative results III

Beyond kinematics includes more features, such as particle-ID and track impact particle-ID and we demonstrate that \mathcal{L} our CNF can accurately model all of these additional features as well. Our generative model for JetClass expands on the versatility of existing jet generation techniques, enhancing their potential utility in high-energy physics research, and o \mathbf{u} more comprehensive understanding of the theories of the

Quality Metrics

Quality of simulation

How well does the generative model describe the training data?

a.u.

a.u.

Two-dimensional metrics

Geant4

 $m_{1,x}$ $m_{1,\,y}$ -0.01 $m_{1,z}$ $-0.12 - 0.01$ $m_{2,x}$ 0.01 0.00 -0.38 1.0 $m_{2, u}$ $0.04 -0.01 -0.29 0.41 1.0$ $m_{2,z}$ 0.07 0.01 -0.34 0.16 0.14 1.0 $E_{\rm vis}$ 0.06 0.02 0.21 -0.08 -0.06 0.08 $E_{\rm inc}$ 0.07 0.01 0.35 0.14 0.10 0.04 0.98 1.00 n_{hit} -0.05 0.02 0.14 -0.03 -0.01 0.18 $0.99 \quad 0.96 \quad 1.0$ $E_1/E_{\rm vis}$ 0.16 0.00 0.93 0.35 0.28 0.42 0.27 0.38 0.21 $E_2/E_{\rm vis}$ -0.13 0.01 0.13 -0.05 -0.06 -0.33 0.23 0.19 0.22 -0.47 $E_3/E_{\rm vis}$ -0.07 -0.01 0.93 -0.35 -0.27 -0.22 0.12 0.28 0.06 $\sqrt{4}$ -0.24 1.0 $E_2/E_{\rm vis}$ $E_{\rm vis}$ $E_{\rm inc}$ $n_{\rm hit}$ $E_1/E_{\rm vis}$ $\begin{matrix} m_{1,\,x}\ m_{1,\,y}\ m_{2,\,x}\ m_{2,\,x}\ m_{2,\,y}\ m_{2,\,y}\ m_{2,\,z}\end{matrix}$ $E_3/E_{\rm v}$

Geant4 - WGAN

Geant4 - BIB-AE PP

Pair-wise correlations contain more information

Multi-dimensional metrics

Capture full phase space information with classifiers

Still depends on training data

(Can also compare multiple models with multi-class classifier and then evaluate on data)

(Or use latent space of pre-trained classifier: Frechet Inception/Particle Distance)

Choice of classifier

How good, is good enough really?

Adding reconstruction number of hits in the right plot is calculated after a cuto \mathbf{r}

Weights

 10^0

 $\begin{array}{c}\n\text{normalized counts} \\
\text{normalized} \\
10^{-1}\n\end{array}$

Quality of simulation

Closing II

Common Datasets that indicate if a constituent is an electron or a photon, o \blacksquare o and the generated incorporated incorporated incorporated incorporate incorporation, and a substantial intera enhancing its potential utility in high-energy physics research.

Common Datasets o and the generated incorporated incorporated incorporated incorporate incorporation, and a substantial intera enhancing its potential utility in high-energy physics research.

 i Jetn j et to facilitate the training of state-of-the-article the-article the-article

The purpose of this challenge is to spur the development and benchmarking of fast and high-fidelity calorimeter shower generation using deep learning methods. Currently, generating calorimeter showers of interacting particles (electrons, photons, pions, ...) using GEANT4 is a major computational bottleneck at the LHC, and it is forecast to overwhelm the computing budget of the LHC experiments
in the near future. Therefore there is an urgent need to develop GEANT4 emulators that are both fast (computationally lightweight) and accurate. The LHC collaborations have been developing fast simulation methods for some time, and the hope of this challenge is to directly compare new deep learning approaches on common benchmarks. It is expected that participants will make use of cutting-edge techniques in generative modeling with deep learning, e.g. GANs, VAEs and normalizing flows.

This challenge is modeled after two previous, highly successful data challenges in HEP - the top tagging community challenge and the LHC Olympics 2020 anomaly detection challenge.

Finally, In size, the size, the size of the size of the size of the kinematics 3 datasets of increasing complexity:

Generative Cheat Sheet

 $\overline{}$

where *w^t* is a standard Wiener process and *f*(*t*) and *g*(*t*) are drift and diffusion coefficients, $\overline{}$

"

!

samples that follow the data distribution. This reverse process is also described by a stochastic