

Representations

Tilman Plehn

Training

Uncertainties

Regression

Representations

Symmetries

Classification

Generation

Reweighting

Extrapolation

Physics-Representation Learning

Tilman Plehn

Universität Heidelberg

AIPHY Geneva, February 2026



Neural network tasks

Similar to fit

- approximate $f_\theta(x) \approx f(x)$
 - $x \in \mathbb{R}^D$ low-dim phase space
 - $f_\theta \in \mathbb{R}$ numerical function
- θ data representation

Probabilities over phase space

- regression $x \rightarrow A_\theta(x)$
- classification $x \rightarrow p_\theta(x)$ [likelihood ratio]
- generation $r \sim \mathcal{N} \rightarrow x \sim p_\theta(x)$
- conditional generation $r \sim \mathcal{N} \rightarrow x \sim p_\theta(x|y)$

LHC representations

- accuracy [checking bias]
 - precision [calibrated uncertainties, coverage]
 - structure [symmetries etc]
- All related by physics knowledge



Likelihood training

Network training as a fit

- learn scalar field $f_{\theta}(x) \approx f(x)$
- statistics: maximize parameter probability given (f_j, σ_j)

$$\theta = \operatorname{argmax} \frac{p(x|\theta)p(\theta)}{p(x)} = \operatorname{argmax} [p(x|\theta)p(\theta)] \simeq \operatorname{argmax} p(x|\theta) .$$

→ maximize tractable likelihood instead [assume Gaussian]

$$p(x|\theta) = \prod_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{|f_j - f_{\theta}(x_j)|^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \mathcal{L} \equiv -\log p(x|\theta) = \sum_j \frac{|f_j - f_{\theta}(x_j)|^2}{2\sigma_j^2} + \operatorname{const}(\theta)$$



Likelihood training

Network training as a fit

- learn scalar field $f_\theta(x) \approx f(x)$
- statistics: maximize parameter probability given (f_j, σ_j)

$$\theta = \operatorname{argmax} \frac{p(x|\theta)p(\theta)}{p(x)} = \operatorname{argmax} [p(x|\theta)p(\theta)] \simeq \operatorname{argmax} p(x|\theta) .$$

→ maximize tractable likelihood instead [assume Gaussian]

$$p(x|\theta) = \prod_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{|f_j - f_\theta(x_j)|^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \mathcal{L} \equiv -\log p(x|\theta) = \sum_j \frac{|f_j - f_\theta(x_j)|^2}{2\sigma_j^2} + \operatorname{const}(\theta)$$

Learned local uncertainty

- uncertainties encoded in training data [statistics, noise, etc]
- Gaussian log-likelihood with normalization

$$\mathcal{L}_{\text{heteroscedastic}} = \frac{|f(x) - f_\theta(x)|^2}{2\sigma_\theta(x)^2} + \log \sigma_\theta(x) + \dots$$

- if needed replace $\sigma_\theta(x)$ by mixture model

→ learn $f_\theta(x)$ and $\sigma_\theta(x)$ together



Likelihood training

Network training as a fit

- learn scalar field $f_\theta(x) \approx f(x)$
- statistics: maximize parameter probability given (f_j, σ_j)

$$\theta = \operatorname{argmax} \frac{p(x|\theta)p(\theta)}{p(x)} = \operatorname{argmax} [p(x|\theta)p(\theta)] \simeq \operatorname{argmax} p(x|\theta) .$$

→ maximize tractable likelihood instead [assume Gaussian]

$$p(x|\theta) = \prod_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{|f_j - f_\theta(x_j)|^2}{2\sigma_j^2}\right)$$

$$\Rightarrow \mathcal{L} \equiv -\log p(x|\theta) = \sum_j \frac{|f_j - f_\theta(x_j)|^2}{2\sigma_j^2} + \operatorname{const}(\theta)$$

Negative log-likelihood minimization

- target function preprocessing

$$f_j \rightarrow \log f_j \quad \text{or} \quad f_j \rightarrow f_j - \langle f_j \rangle \quad \text{or} \quad f_j \rightarrow \frac{f_j}{\langle f_j \rangle} \dots$$

- challenging dimensionality of network parameters θ [learning rate α]

$$\theta_j^{t+1} = \theta_j^t - \alpha \left\langle \frac{\partial \mathcal{L}^t}{\partial \theta_j} \right\rangle = \alpha \left\langle \nabla_{\theta^t} \log p(\theta^t | x_{\text{train}}) \right\rangle$$

→ Naive scheduled minimization, by physics standards



1- Repulsive ensembles

Posterior from network ensemble

- ensemble training with repulsive force

$$\theta^{t+1} = \theta^t + \alpha \nabla_{\theta^t} \left[\log p(\theta^t | x_{\text{train}}) - \frac{1}{n} \sum_{j=1}^n \tilde{k}(\theta^t, \theta_j^t) \right].$$

- trajectories vs continuity equation

$$\frac{d\theta}{dt} = v(\theta, t) \quad \Leftrightarrow \quad \frac{\partial \rho(\theta, t)}{\partial t} = -\nabla_{\theta} [v(\theta, t) \rho(\theta, t)]$$

- Fokker-Planck equation

$$\frac{d\theta}{dt} \simeq \frac{\theta^{t+1} - \theta^t}{\alpha} = -\nabla_{\theta^t} \log \frac{\rho(\theta^t)}{\pi(\theta^t)}$$

- continuity equation defining stationary solution $\rho(\theta, t) = \pi(\theta)$

$$\begin{aligned} \frac{\partial \rho(\theta, t)}{\partial t} &= \nabla_{\theta} \left[\rho(\theta, t) \nabla_{\theta} \log \frac{\rho(\theta, t)}{\pi(\theta)} \right] \\ &= \nabla_{\theta} \left[\rho(\theta, t) \frac{\nabla_{\theta} \rho(\theta, t)}{\rho(\theta, t)} \right] - \nabla_{\theta} [\rho(\theta, t) \nabla_{\theta} \log \pi(\theta)] \\ &= \nabla_{\theta}^2 \rho(\theta, t) - \nabla_{\theta} [\rho(\theta, t) \nabla_{\theta} \log \pi(\theta)] \stackrel{\rho \rightarrow \pi}{=} 0 \end{aligned}$$



1- Repulsive ensembles

Posterior from network ensemble

- ensemble training with repulsive force

$$\theta^{t+1} = \theta^t + \alpha \nabla_{\theta^t} \left[\log p(\theta^t | x_{\text{train}}) - \frac{1}{n} \sum_{j=1}^n \tilde{k}(\theta^t, \theta_j^t) \right].$$

- trajectories vs continuity equation

$$\frac{d\theta}{dt} = v(\theta, t) \quad \Leftrightarrow \quad \frac{\partial \rho(\theta, t)}{\partial t} = -\nabla_{\theta} [v(\theta, t) \rho(\theta, t)]$$

- Fokker-Planck equation

$$\frac{d\theta}{dt} \simeq \frac{\theta^{t+1} - \theta^t}{\alpha} = -\nabla_{\theta^t} \log \frac{\rho(\theta^t)}{\pi(\theta^t)}$$

- training ODE with kernel for ρ

$$\begin{aligned} \frac{\theta^{t+1} - \theta^t}{\alpha} &= \nabla_{\theta^t} [\log \pi(\theta^t) - \log \rho(\theta^t)] \\ &= \nabla_{\theta^t} \log \pi(\theta^t) - \nabla_{\theta^t} \log \left[\frac{1}{n} \sum_j k(\theta^t, \theta_j^t) \right] \\ &\equiv \nabla_{\theta^t} \log p(\theta | x_{\text{train}}^t) - \frac{\nabla_{\theta^t} \sum_j k(\theta^t, \theta_j^t)}{\sum_j k(\theta^t, \theta_j^t)} \end{aligned}$$



1- Repulsive ensembles

Posterior from network ensemble

- ensemble training with repulsive force

$$\theta^{t+1} = \theta^t + \alpha \nabla_{\theta^t} \left[\log p(\theta^t | x_{\text{train}}) - \frac{1}{n} \sum_{j=1}^n \tilde{k}(\theta^t, \theta_j^t) \right].$$

- trajectories vs continuity equation

$$\frac{d\theta}{dt} = v(\theta, t) \quad \Leftrightarrow \quad \frac{\partial \rho(\theta, t)}{\partial t} = -\nabla_{\theta} [v(\theta, t) \rho(\theta, t)]$$

- Fokker-Planck equation

$$\frac{d\theta}{dt} \simeq \frac{\theta^{t+1} - \theta^t}{\alpha} = -\nabla_{\theta^t} \log \frac{\rho(\theta^t)}{\pi(\theta^t)}$$

- training ODE with kernel for ρ

$$\begin{aligned} \frac{\theta^{t+1} - \theta^t}{\alpha} &= \nabla_{\theta^t} [\log \pi(\theta^t) - \log \rho(\theta^t)] \\ &= \nabla_{\theta^t} \log \pi(\theta^t) - \nabla_{\theta^t} \log \left[\frac{1}{n} \sum_j k(\theta^t, \theta_j^t) \right] \\ &\equiv \nabla_{\theta^t} \log p(\theta | x_{\text{train}}^t) - \beta \frac{\nabla_{\theta^t} \sum_j k(\theta^t, \theta_j^t)}{\sum_j k(\theta^t, \theta_j^t)} \equiv -\nabla_{\theta^t} \mathcal{L}_{\text{RE}} \end{aligned}$$

→ Joint ensemble training



1- Repulsive ensembles

Posterior from network ensemble

- ensemble training with repulsive force

$$\theta^{t+1} = \theta^t + \alpha \nabla_{\theta^t} \left[\log p(\theta^t | x_{\text{train}}) - \frac{1}{n} \sum_{j=1}^n \tilde{k}(\theta^t, \theta_j^t) \right].$$

- trajectories vs continuity equation

$$\frac{d\theta}{dt} = v(\theta, t) \quad \Leftrightarrow \quad \frac{\partial \rho(\theta, t)}{\partial t} = -\nabla_{\theta} [v(\theta, t) \rho(\theta, t)]$$

- Fokker-Planck equation

$$\frac{d\theta}{dt} \simeq \frac{\theta^{t+1} - \theta^t}{\alpha} = -\nabla_{\theta^t} \log \frac{\rho(\theta^t)}{\pi(\theta^t)}$$

→ Joint ensemble training

Repulsive ensembles

- train network ensemble
- apply repulsive force
 - kernel in function space
 - test impact of kernel (?)

→ Alternative for 'statistical' uncertainty



2- Bayesian networks

Learned function statistically

- amplitude A over phase phase

$$\langle A \rangle = \int dA A p(A)$$

- internal representation θ of training data T [think Gaussian with mean and width]

$$p(A) = \int d\theta p(A|\theta) p(\theta|T)$$

→ θ -distribution defining Bayesian NN



2- Bayesian networks

Learned function statistically

- amplitude A over phase phase

$$\langle A \rangle = \int dA A p(A)$$

- internal representation θ of training data T [think Gaussian with mean and width]

$$p(A) = \int d\theta p(A|\theta) p(\theta|T)$$

→ θ -distribution defining Bayesian NN

Variational approximation

- definition of training

$$p(A) = \int d\theta p(A|\theta) p(\theta|T) \approx \int d\theta p(A|\theta) q(\theta)$$

- similarity through minimal KL-divergence [Bayes' theorem to remove unknown posterior]

$$\begin{aligned} D_{\text{KL}}[q(\theta), p(\theta|T)] &= \int d\theta q(\theta) \log \frac{q(\theta)}{p(\theta|T)} \\ &= \int d\theta q(\theta) \log \frac{q(\theta)p(T)}{p(T|\theta)p(\theta)} \\ &\approx D_{\text{KL}}[q(\theta), p(\theta)] - \int d\theta q(\theta) \log p(T|\theta) \equiv \mathcal{L} \end{aligned}$$

→ Two-term loss: likelihood + prior



2- Bayesian networks

Statistical network evaluation

- expectation value from network $q(\theta)$

$$\begin{aligned}\langle A \rangle &= \int dA d\theta \ A \ p(A|\theta) \ q(\theta) \\ &\equiv \int d\theta \ q(\theta) \bar{A}(\theta) \quad \text{with} \quad \bar{A}(\theta) = \int dA \ A \ p(A|\theta)\end{aligned}$$

- corresponding variance

$$\begin{aligned}\sigma_{\text{tot}}^2 &= \int dA d\theta \ (A - \langle A \rangle)^2 \ p(A|\theta) \ q(\theta) \\ &= \int d\theta \ q(\theta) \left[\overline{A^2}(\theta) - \bar{A}(\theta)^2 + (\bar{A}(\theta) - \langle A \rangle)^2 \right] \equiv \sigma_{\text{syst}}^2 + \sigma_{\text{stat}}^2\end{aligned}$$

Two uncertainties

- statistical — vanishing for perfect training: $q(\theta) \rightarrow \delta(\theta - \theta_0)$

$$\sigma_{\text{stat}}^2 = \int d\theta \ q(\theta) \left[\bar{A}(\theta) - \langle A \rangle \right]^2$$

- systematic — vanishing for perfect data: $p(A|\theta) \rightarrow \delta(A - A_0)$

$$\sigma_{\text{syst}}^2 = \int d\theta \ q(\theta) \left[\overline{A^2}(\theta) - \bar{A}(\theta)^2 \right]$$

- (ir)-reducible errors in ML language

→ Systematics dominant for LHC



Regularization

- BNN loss

$$\mathcal{L} = - \int d\theta \, q(\theta) \log p(T|\theta) + D_{\text{KL}}[q(\theta), p(\theta)]$$



2- Bayesian vs deterministic networks

Regularization

- Gaussian prior

$$\mathcal{L} = - \int d\theta \, q(\theta) \log p(T|\theta) + \frac{\sigma_q^2 - \sigma_p^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} + \dots$$

- deterministic network

$$q(\theta) = \delta(\theta - \theta_0) \quad \Rightarrow \quad \mathcal{L} \approx -\log p(T|\theta_0) + \frac{(\theta_0 - \mu_p)^2}{2\sigma_p^2}$$

→ Likelihood with L2-regularization



2- Bayesian vs deterministic networks

Regularization

- Gaussian prior

$$\mathcal{L} = - \int d\theta \, q(\theta) \log p(T|\theta) + \frac{\sigma_q^2 - \sigma_p^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} + \dots$$

- deterministic network

$$q(\theta) = \delta(\theta - \theta_0) \quad \Rightarrow \quad \mathcal{L} \approx -\log p(T|\theta_0) + \frac{(\theta_0 - \mu_p)^2}{2\sigma_p^2}$$

→ Likelihood with L2-regularization

Dropout

- Bernoulli weights

$$q(\theta) \rightarrow q(x) = \rho^x (1 - \rho)^{1-x} \Big|_{x=0,1} \quad \text{with } \theta = x\theta_0$$

- likelihood loss

$$\mathcal{L} = - \sum_{x=0,1} \left[\rho^x (1 - \rho)^{1-x} \right] \log p(T|x\theta_0) = -\rho \log p(T|\theta_0)$$

- likelihood Gaussian or whatever else...

→ Regularized likelihood with dropout



3- Evidential regression

Uncertainties from latent distribution, without sampling

- evidential distribution in variational inference

$$p(A) = \int d\bar{A} d\sigma^2 p(A|\bar{A}, \sigma^2) p(\bar{A}, \sigma^2 | T) \approx \int d\bar{A} d\sigma^2 p(A|\bar{A}, \sigma^2) q(\bar{A}, \sigma^2)$$

assuming $p(A|\bar{A}, \sigma^2) = \mathcal{N}(A|\bar{A}, \sigma^2)$ with learned $(\bar{A}, \sigma^2)$

- choose $q(\bar{A}, \sigma^2 | \theta_0)$ as conjugate prior for Gaussian likelihood

$$q_m(\bar{A}, \sigma^2) = \frac{\beta^\alpha \sqrt{v}}{\Gamma(\alpha) \sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left(-\frac{2\beta + v(\gamma - \bar{A})^2}{2\sigma^2}\right)$$

with $m = \{\gamma, v, \alpha, \beta\}(\theta)$

- analytical statistical properties

$$\langle A \rangle = \int dA A p(A)$$

$$= \int d\bar{A} d\sigma^2 \bar{A} q_m(\bar{A}, \sigma^2) = \gamma$$

$$\sigma_{\text{syst}}^2 = \int d\bar{A} d\sigma^2 \sigma^2 q_m(\bar{A}, \sigma^2) = \frac{\beta}{\alpha - 1}$$

$$\sigma_{\text{stat}}^2 = \int d\bar{A} d\sigma^2 [\bar{A} - \langle A \rangle]^2 q_m(\bar{A}, \sigma^2) = \frac{\beta}{v(\alpha - 1)}$$

- only 3 of 4 parameters statistically fixed
- generalizable to Gaussian mixture model



3- Evidential regression

Uncertainties from latent distribution, without sampling

- evidential distribution in variational inference

$$p(A) = \int d\bar{A} d\sigma^2 p(A|\bar{A}, \sigma^2) p(\bar{A}, \sigma^2 | T) \approx \int d\bar{A} d\sigma^2 p(A|\bar{A}, \sigma^2) q(\bar{A}, \sigma^2)$$

$$\text{assuming } p(A|\bar{A}, \sigma^2) = \mathcal{N}(A|\bar{A}, \sigma^2) \quad \text{with learned } (\bar{A}, \sigma^2)$$

- choose $q(\bar{A}, \sigma^2 | \theta_0)$ as conjugate prior for Gaussian likelihood

$$q_m(\bar{A}, \sigma^2) = \frac{\beta^\alpha \sqrt{v}}{\Gamma(\alpha) \sqrt{2\pi\sigma^2}} \left(\frac{1}{\sigma^2} \right)^{\alpha+1} \exp\left(-\frac{2\beta + v(\gamma - \bar{A})^2}{2\sigma^2} \right)$$

$$\text{with } m = \{\gamma, v, \alpha, \beta\}(\theta)$$

- only 3 of 4 parameters statistically fixed
- analytical statistical properties
- analytical likelihood

$$\begin{aligned} p(A) &= \int d\bar{A} d\sigma^2 \mathcal{N}(A|\bar{A}, \sigma^2) q_m(\bar{A}, \sigma^2) \\ &= \text{St} \left(A \middle| \gamma, \frac{\beta(1+v)}{v\alpha}, 2\alpha \right) \end{aligned}$$

- generalizable to Gaussian mixture model

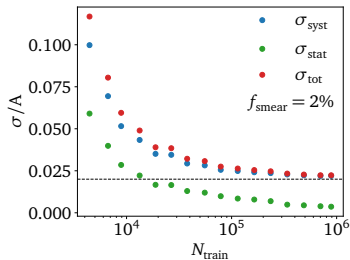
→ Another alternative for learned uncertainty



Amplitudes with calibrated uncertainties

Amplitude regression

- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau

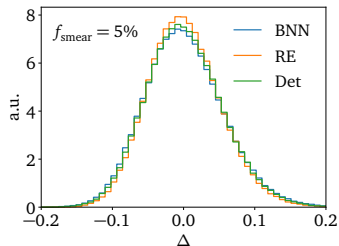
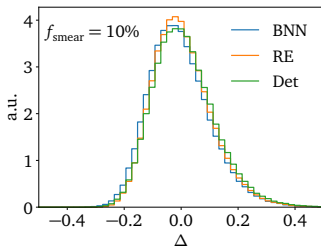


Amplitudes with calibrated uncertainties

Amplitude regression

- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau
- accuracy over phase space

$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$



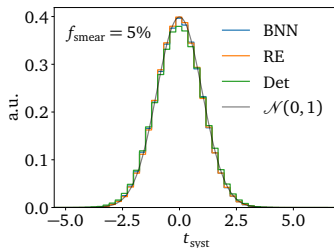
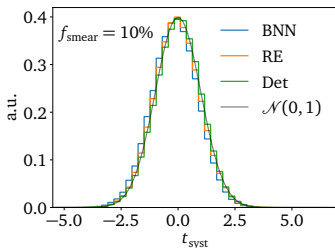
Amplitude regression

- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau
- accuracy over phase space

$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$

- pull over phase space

$$t_{\text{syst}}(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{\sigma_{\text{syst}}(x)}$$



Amplitudes with calibrated uncertainties

Amplitude regression

- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau
- accuracy over phase space
- pull over phase space

$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$

$$t_{\text{syst}}(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{\sigma_{\text{syst}}(x)}$$

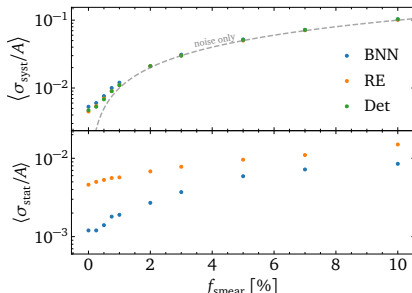
Towards zero noise

- scaling

$$\sigma_{\text{syst}}^2 - \sigma_{\text{syst},0}^2 \approx \sigma_{\text{train}}^2$$

- plateau $\langle \sigma_{\text{syst}}/A \rangle \sim 0.4\%$

→ **Limiting factor??**



Amplitudes with calibrated uncertainties

Amplitude regression

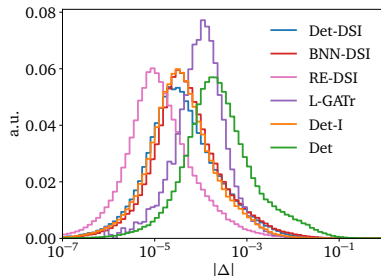
- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau
- accuracy over phase space
- pull over phase space

$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$

$$t_{\text{syst}}(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{\sigma_{\text{syst}}(x)}$$

Data representation

- amplitude from invariants
- learn Minkowski metric
- Deep-sets-invariant network



Amplitudes with calibrated uncertainties

Amplitude regression

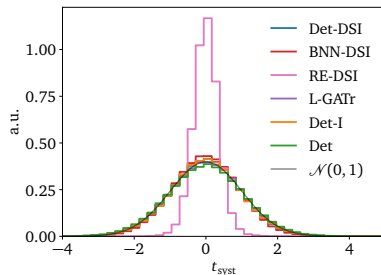
- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau
- accuracy over phase space
- pull over phase space

$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$

$$t_{\text{syst}}(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{\sigma_{\text{syst}}(x)}$$

Data representation

- amplitude from invariants
 - learn Minkowski metric
 - Deep-sets-invariant network
- **Calibrated systematics**



Amplitudes with calibrated uncertainties

Amplitude regression

- loop amplitude over phase space $gg \rightarrow \gamma\gamma g(g)$
- systematics: **artificial noise**
- statistics plateau
- accuracy over phase space
- pull over phase space

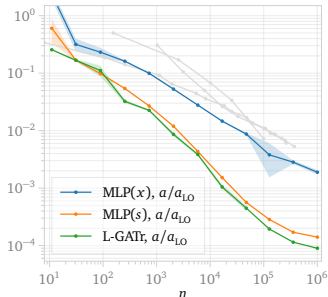
$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$

$$t_{\text{syst}}(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{\sigma_{\text{syst}}(x)}$$

Data representation

- amplitude from invariants
 - learn Minkowski metric
 - Deep-sets-invariant network
- **On to real loop integrals**

Approximation error of f_4



Phase space gaps

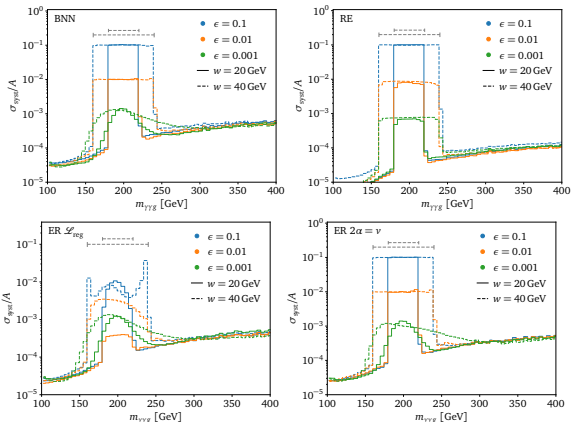
Noisy amplitude gap

- step-function relative noise in $m_{\gamma\gamma g}$ gap

$$A_{\text{train}}(x) = \begin{cases} \mathcal{N}(A_{\text{true}}(x), \epsilon A_{\text{true}}(x)) & |m_{\gamma\gamma g}(x) - m_{\text{thresh}}| < w \\ A_{\text{true}} & |m_{\gamma\gamma g}(x) - m_{\text{thresh}}| \geq w \end{cases}$$

- compare Bayesian NN, ensembles, evidential regression

→ Only little noise corrected, any noise learned



Latent representation: transformer

Non-local correlations of point clouds

- construct D-dimensional numerical representation $x \rightarrow z$
- start with (compact) **query** representation

$$x_i \longrightarrow q = \frac{x_i}{|x|}$$

- orthonormal **values** basis [related to q through scalar product]

$$q = \sum_j (q \cdot v_j) v_j$$

- simpler orthogonal **keys** basis

$$q = \sum_j (q \cdot k_j) v_j \quad \text{with} \quad k_j = \frac{v_j}{v^2}$$

→ **Permutation-equivariant self-attention representation**

$$x_i \longrightarrow z_i = \sum_j (q_i \cdot k_j) v_j$$



Latent representation: transformer

Non-local correlations of point clouds

- construct D-dimensional numerical representation $x \rightarrow z$
- start with (compact) **query** representation

$$x_i \rightarrow q = \frac{x_i}{|x|}$$

- orthonormal **values** basis [related to q through scalar product]

$$q = \sum_j (q \cdot v_j) v_j$$

- simpler orthogonal **keys** basis

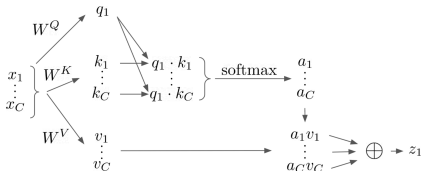
$$q = \sum_j (q \cdot k_j) v_j \quad \text{with} \quad k_j = \frac{v_j}{v^2}$$

→ **Permutation-equivariant self-attention representation**

$$x_i \rightarrow z_i = \sum_j (q_i \cdot k_j) v_j$$

In terms of learnable matrices

- latent query $q = W^Q x$
- latent key $k = W^K x$
- correlation $A_{ij} = q_i \cdot k_j$
- latent value $v = W^V x$
- constructed $z = A v$



Equivariant latent representation

Encode more known symmetries

- Lorentz co-variance/equivariance $f_\theta(S(x)) = S(f_\theta(x))$
- input: 4-vectors vs Mandelstams $\rightarrow$ scalars, vectors, etc?
- geometric product extending vector space

$$xy = \frac{\{x, y\}}{2} + \frac{[x, y]}{2} \quad \text{with} \quad \{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu}$$

- metric reducing, bivector increasing grade

$$\gamma^\mu \gamma^\nu = \frac{\{\gamma^\mu, \gamma^\nu\}}{2} + \frac{[\gamma^\mu, \gamma^\nu]}{2} \equiv g^{\mu\nu} + \sigma^{\mu\nu}$$

- multivector of geometric algebra [just like supersymmetry in 90s]

$$x = x^S 1 + x_\mu^V \gamma^\mu + x_{\mu\nu}^B \sigma^{\mu\nu} + x_\mu^A \gamma^\mu \gamma^5 + x^P \gamma^5 \quad \text{with} \quad \begin{pmatrix} x^S \\ x_\mu^V \\ x_{\mu\nu}^B \\ x_\mu^A \\ x^P \end{pmatrix} \in \mathbb{R}^{16}$$

- example input (PID, p_μ)

$$x^S = \text{PID} \quad x_\mu^V = p_\mu \quad x_{\mu\nu}^T = x_\mu^A = x^P = 0$$

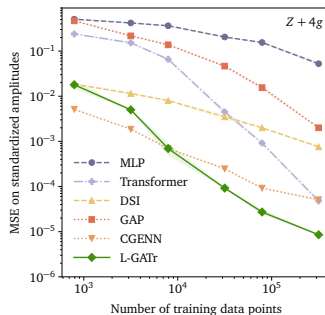
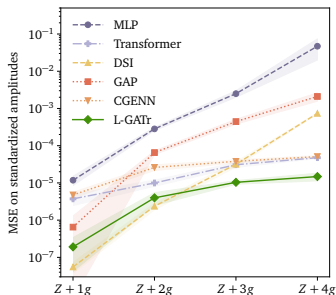
$\rightarrow$ Organize network layers by grade: L-GATr



L-GATr amplitudes

Performance is all you need

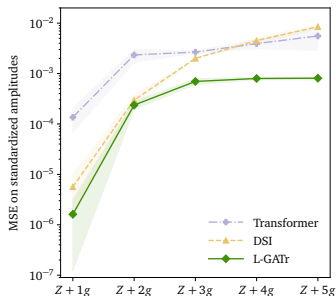
- LO transition amplitudes $q\bar{q} \rightarrow Z + 1\dots 4 g$
- amplitude regression: cost scaling with dimensionality
→ low-dimensional representation better
- performance scaling with multiplicity and training data
→ winning transformers



L-GATr amplitudes

Performance is all you need

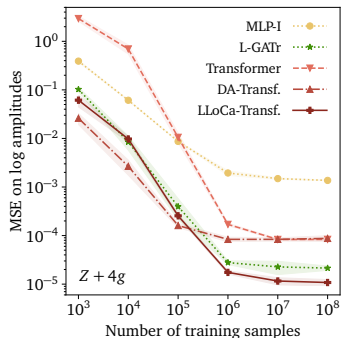
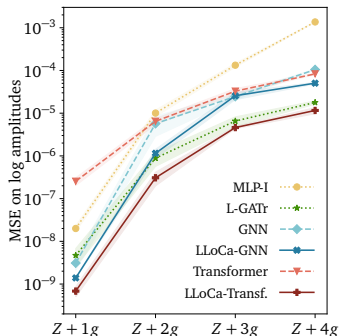
- LO transition amplitudes $q\bar{q} \rightarrow Z + 1\dots 4 g$
 - amplitude regression: cost scaling with dimensionality
→ low-dimensional representation better
 - performance scaling with multiplicity and training data
→ winning transformers
 - more scaling with less data
→ equivariance with additional advantage
- **Equvariant transformers current winner**



LLoCa vs L-GATr amplitudes

Alternative implementation

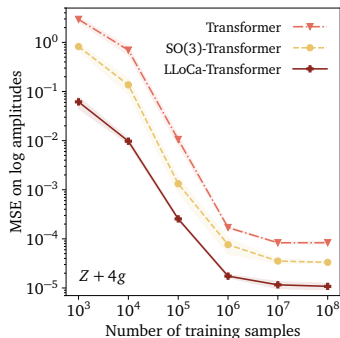
- Better performance is all you need
- local Lorentz frame for each particle
message passing between particles in local frames
- scaling with multiplicity and training data
→ L-GATs and LLoCa comparable



LLoCa vs L-GATr amplitudes

Alternative implementation

- Better performance is all you need
 - local Lorentz frame for each particle
message passing between particles in local frames
 - scaling with multiplicity and training data
→ L-GATs and LLoCa comparable
 - broken symmetries: rotations
→ intermediate symmetries means intermediate performance
- Advantage of equivariance for all implementations



Classification

Learn probabilities

- standard first step of analysis: identify objects optimally
- learn signal and background probabilities

$$D_{\text{KL}}[p_{\text{data}}, p_{\text{model}}] = \left\langle \log \frac{p_{\text{data}}}{p_{\text{model}}} \right\rangle_{p_{\text{data}}} \equiv \int dx \, p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)}.$$

- start with signal and background probabilities

$$\begin{aligned} \mathcal{L}_{\text{class}} &= \sum_{j=S,B} D_{\text{KL}}[p_{\text{data},j}, p_{\text{model},j}] \\ &= \left\langle \log p_{\text{data},S} - \log p_{\text{model},S} \right\rangle_{p_{\text{data},S}} + \left\langle \log p_{\text{data},B} - \log p_{\text{model},B} \right\rangle_{p_{\text{data},B}} \\ &= - \left\langle \log p_{\text{model},S} \right\rangle_{p_{\text{data},S}} - \left\langle \log p_{\text{model},B} \right\rangle_{p_{\text{data},B}} + \text{const}(\theta) \\ \Rightarrow \quad \mathcal{L}_{\text{class}} &= - \sum_{\{x\}} \left[p_{\text{data},S} \log p_{\text{model},S} + p_{\text{data},B} \log p_{\text{model},B} \right] \end{aligned}$$

- perfect training

$$\begin{aligned} 0 &\stackrel{!}{=} - \frac{\delta}{\delta p_{\text{model},S}} \sum_{\{x\}} \left[p_{\text{data},S} \log p_{\text{model},S} + (1 - p_{\text{data},S}) \log(1 - p_{\text{model},S}) \right] \\ &= - \sum_{\{x\}} \left[\frac{p_{\text{data},S}}{p_{\text{model},S}} - \frac{1 - p_{\text{data},S}}{1 - p_{\text{model},S}} \right] \quad \Leftrightarrow \quad p_{\text{data},S} = p_{\text{model},S} \end{aligned}$$



Classification

Learn probabilities

- standard first step of analysis: identify objects optimally
- learn signal and background probabilities

$$D_{\text{KL}}[p_{\text{data}}, p_{\text{model}}] = \left\langle \log \frac{p_{\text{data}}}{p_{\text{model}}} \right\rangle_{p_{\text{data}}} \equiv \int dx \, p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)}.$$

- start with signal and background probabilities

$$\begin{aligned} \mathcal{L}_{\text{class}} &= \sum_{j=S,B} D_{\text{KL}}[p_{\text{data},j}, p_{\text{model},j}] \\ &= \left\langle \log p_{\text{data},S} - \log p_{\text{model},S} \right\rangle_{p_{\text{data},S}} + \left\langle \log p_{\text{data},B} - \log p_{\text{model},B} \right\rangle_{p_{\text{data},B}} \\ &= -\left\langle \log p_{\text{model},S} \right\rangle_{p_{\text{data},S}} - \left\langle \log p_{\text{model},B} \right\rangle_{p_{\text{data},B}} + \text{const}(\theta) \\ \Rightarrow \quad \mathcal{L}_{\text{class}} &= -\sum_{\{x\}} \left[p_{\text{data},S} \log p_{\text{model},S} + p_{\text{data},B} \log p_{\text{model},B} \right] \end{aligned}$$

- relation to Bernoulli distribution

$$\begin{aligned} \mathcal{L}_{\text{class}} &= -\sum_{\{x\}} \left[p_{\text{data},S} \log p_{\text{model},S} + (1 - p_{\text{data},S}) \log(1 - p_{\text{model},S}) \right] \\ &= -\sum_{\{x\}} \log \left[p_{\text{model},S}^{p_{\text{data},S}} (1 - p_{\text{model},S})^{1-p_{\text{data},S}} \right] \end{aligned}$$



Classification

Learn probabilities

- standard first step of analysis: identify objects optimally
- start with signal and background probabilities

$$\mathcal{L}_{\text{class}} = - \sum_{\{x\}} \left[p_{\text{data},S} \log p_{\text{model},S} + p_{\text{data},B} \log p_{\text{model},B} \right]$$

Neyman-Pearson lemma

- classifier

$$D(x) \in [0, 1] \rightarrow \begin{cases} 0 & \text{background} \\ 1 & \text{signal} \end{cases}$$

- loss

$$\begin{aligned} \mathcal{L}_{\text{class}} &= \sum_{\{x\}} \left[p_{\text{data},S}(x)(1 - D(x)) + p_{\text{data},B}(x)D(x) \right] \\ &\sim - \sum_{\{x\}} \left[p_{\text{data},S}(x) \log D(x) + p_{\text{data},B}(x) \log(1 - D(x)) \right] \end{aligned}$$

- loss minimum: Neyman-Pearson lemma

$$\frac{\delta}{\delta D} \left[p_{\text{data},S}(x) \log D + p_{\text{data},B}(x) \log(1 - D) \right] = \frac{p_{\text{data},S}(x)}{D} - \frac{p_{\text{data},B}(x)}{1 - D} \stackrel{!}{=} 0$$

$$\Leftrightarrow D(x) p_{\text{data},B}(x) = (1 - D(x)) p_{\text{data},S}(x) \quad \Leftrightarrow D(x) = \frac{p_{\text{data},S}(x)}{p_{\text{data},S}(x) + p_{\text{data},B}(x)}$$

→ **Classifying = density ratio estimation**



Classification without labels

Classifier training on mixed samples

- signal ratios f_1 and f_2

$$\begin{aligned} \begin{pmatrix} p_1(x) \\ p_2(x) \end{pmatrix} &= \begin{pmatrix} f_1 & 1 - f_1 \\ f_2 & 1 - f_2 \end{pmatrix} \begin{pmatrix} p_S(x) \\ p_B(x) \end{pmatrix} \\ \Leftrightarrow \begin{pmatrix} p_S(x) \\ p_B(x) \end{pmatrix} &= \frac{1}{f_1 - f_2} \begin{pmatrix} 1 - f_2 & f_1 - 1 \\ -f_2 & f_1 \end{pmatrix} \begin{pmatrix} p_1(x) \\ p_2(x) \end{pmatrix} \end{aligned}$$

- trainable likelihood ratio

$$\frac{p_1(x)}{p_2(x)} = \frac{f_1 p_S(x) + (1 - f_1) p_B(x)}{f_2 p_S(x) + (1 - f_2) p_B(x)} = \frac{f_1 \frac{p_S(x)}{p_B(x)} + 1 - f_1}{f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2}$$

- relation to signal-background ratio

$$\frac{d}{d(p_S/p_B)} \frac{p_1(x)}{p_2(x)} = \frac{f_1 \left[f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2 \right] - f_2 \left[f_1 \frac{p_S(x)}{p_B(x)} + 1 - f_1 \right]}{\left[f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2 \right]^2} = \frac{f_1 - f_2}{\left[f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2 \right]^2}$$

→ Monotonuous function for working points [fix calibration]



Classification without labels

Classifier training on mixed samples

- signal ratios f_1 and f_2

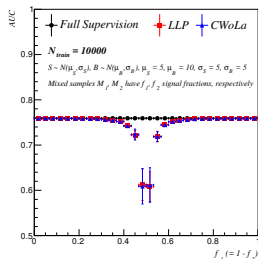
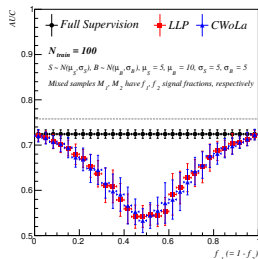
$$\begin{pmatrix} p_1(x) \\ p_2(x) \end{pmatrix} = \begin{pmatrix} f_1 & 1 - f_1 \\ f_2 & 1 - f_2 \end{pmatrix} \begin{pmatrix} p_S(x) \\ p_B(x) \end{pmatrix}$$

$$\Leftrightarrow \begin{pmatrix} p_S(x) \\ p_B(x) \end{pmatrix} = \frac{1}{f_1 - f_2} \begin{pmatrix} 1 - f_2 & f_1 - 1 \\ -f_2 & f_1 \end{pmatrix} \begin{pmatrix} p_1(x) \\ p_2(x) \end{pmatrix}$$

- relation to signal-background ratio

$$\frac{d}{d(p_S/p_B)} \frac{p_1(x)}{p_2(x)} = \frac{f_1 \left[f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2 \right] - f_2 \left[f_1 \frac{p_S(x)}{p_B(x)} + 1 - f_1 \right]}{\left[f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2 \right]^2} = \frac{f_1 - f_2}{\left[f_2 \frac{p_S(x)}{p_B(x)} + 1 - f_2 \right]^2}$$

→ Monotonuous function for working points [fix calibration]



Generative networks

Generative networks

- start with training sample over x
 - estimate density $p_{\text{model}}(x)$
 - sample $x \sim p_{\text{model}}(x)$ from, e.g. from Gaussian $r \sim \mathcal{N}(r)$
- [Learn Jacobian](#)



Generative networks

Generative networks

- start with training sample over x
 - estimate density $p_{\text{model}}(x)$
 - sample $x \sim p_{\text{model}}(x)$ from, e.g. from Gaussian $r \sim \mathcal{N}(r)$
- Learn Jacobian

Normalizing flow

- bijective mapping

$$\text{latent } r \sim p_{\text{latent}} \quad \begin{array}{c} \xleftarrow{G_{\theta}(r) \rightarrow} \\ \xleftarrow{\bar{G}_{\theta}(x)} \end{array} \quad \text{phase space } x \sim p_{\text{data}}$$

- tractable Jacobian

$$dx \, p_{\text{model}}(x) = dr \, p_{\text{latent}}(r)$$

$$p_{\text{model}}(x) = p_{\text{latent}}(\bar{G}_{\theta}(x)) \left| \frac{\partial \bar{G}_{\theta}(x)}{\partial x} \right|$$

- likelihood loss

$$\mathcal{L}_{\text{INN}} = -\left\langle \log p_{\text{model}}(x) \right\rangle_{p_{\text{data}}} = -\left\langle \log p_{\text{latent}}(\bar{G}_{\theta}(x)) + \log \left| \frac{\partial \bar{G}_{\theta}(x)}{\partial x} \right| \right\rangle_{p_{\text{data}}}$$

⇒ Learn Jacobian bi-directionally



Uncertainty-aware generators

Unsupervised Bayesian networks

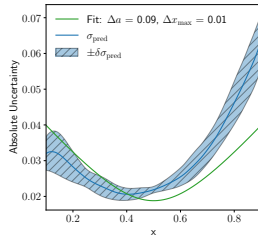
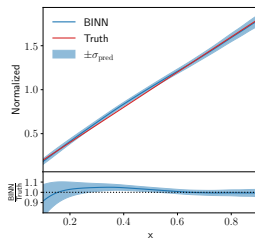
- data: event sample [points in 2D space]
learn phase space density
normalizing flow mapping to latent space [INN]
standard distribution in latent space [Gaussian]
mapping bijective
sample from latent space
- Bayesian version
allow weight distributions
learn uncertainty map
- 2D wedge ramp

$$p(x) = ax + b = ax + \frac{1 - \frac{a}{2}(x_{\max}^2 - x_{\min}^2)}{x_{\max} - x_{\min}}$$

$$(\Delta p)^2 = \left(x - \frac{1}{2}\right)^2 (\Delta a)^2 + \left(1 + \frac{a}{2}\right)^2 (\Delta x_{\max})^2 + \left(1 - \frac{a}{2}\right)^2 (\Delta x_{\min})^2$$

explaining minimum in $\sigma_{\text{pred}}(x)$

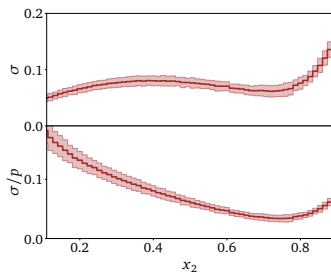
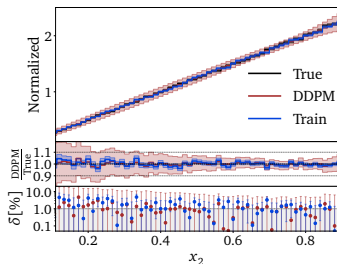
→ INNs just (non-parametric) fits



More architectures

Alternative architectures

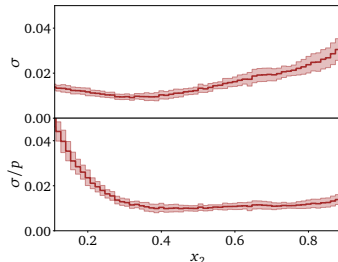
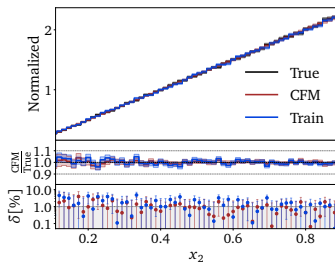
- always a fit?
- expressivity vs architecture?
- discrete diffusion model



More architectures

Alternative architectures

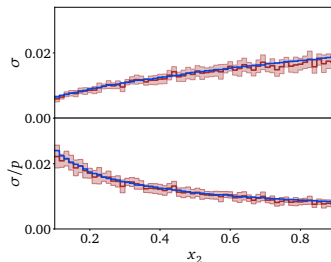
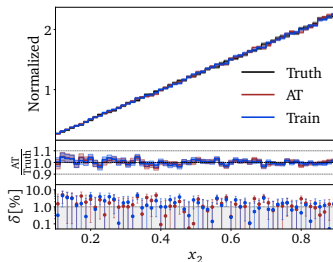
- always a fit?
- expressivity vs architecture?
- discrete diffusion model
- continuous diffusion model



More architectures

Alternative architectures

- always a fit?
- expressivity vs architecture?
- discrete diffusion model
- continuous diffusion model
- autoregressive transformer (bins)



More architectures

Alternative architectures

- always a fit?
- expressivity vs architecture?
- discrete diffusion model
- continuous diffusion model
- autoregressive transformer (bins)

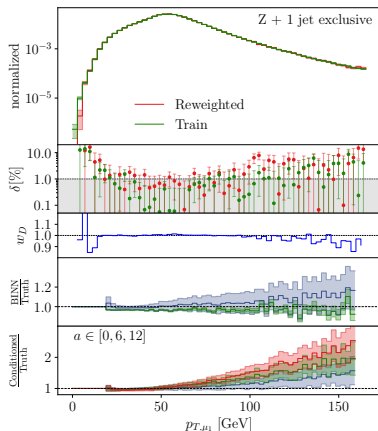
Z+jets events

- per-cent accuracy on density
- statistical uncertainty from BNN
- systematics in weighted training data

$$w = 1 + a \left(\frac{p_{T,j_1} - 15 \text{ GeV}}{100 \text{ GeV}} \right)^2$$

- training conditional on a sampling $\{r, a\}$

→ Precision event generation



Testing generative networks

Compare network to training/test data

- supervised: histogram deviation from truth
- unsupervised density → histogram discriminator

$$w(x_i) = \frac{D(x_i)}{1 - D(x_i)} = \frac{p_{\text{data}}(x_i)}{p_{\text{model}}(x_i)}$$

→ Using interpretable phase space



Testing generative networks

Compare network to training/test data

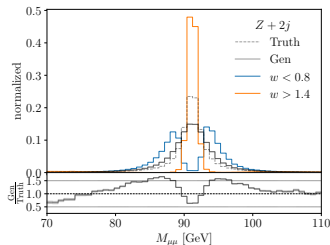
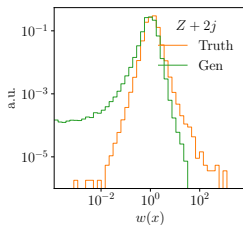
- supervised: histogram deviation from truth
- unsupervised density $\rightarrow$ histogram discriminator

$$w(x_i) = \frac{D(x_i)}{1 - D(x_i)} = \frac{p_{\text{data}}(x_i)}{p_{\text{model}}(x_i)}$$

$\rightarrow$ Using interpretable phase space

Applied to LHC events over phase space

- shape and width of w -histogram
- pattern in (interpretable) phase space?



$\rightarrow$ Local failure modes



Equivariant event generation

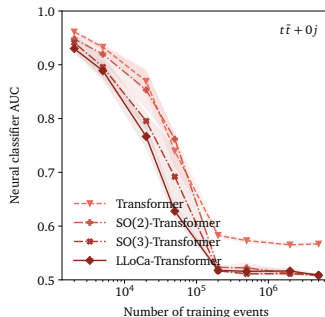
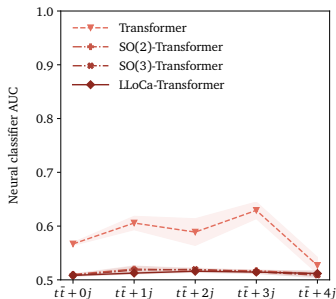
State of the art

- conditional flow matching or continuous normalizing flow
- encode velocity using neural network

$$\frac{d\theta}{dt} = v(\theta, t) \quad \Leftrightarrow \quad \frac{\partial \rho(\theta, t)}{\partial t} = -\nabla_{\theta} [v(\theta, t) \rho(\theta, t)]$$

- transformer for potentially non-local correlations
- equivariant transformer for efficiency
- broken Lorentz symmetry by beam pipe

→ Precision-phase space a solved problem



Generative amplification

How many events can I sample from a network trained on n events?

- learned phase space density

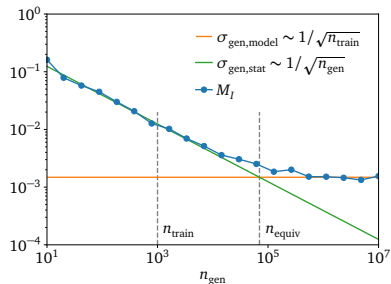
$$D_{\text{true}}^{n_{\text{train}}} \sim p_{\text{true}} \approx p_{\text{gen}}(x) \quad \Rightarrow \quad p_{\text{gen}}(x) \stackrel{?}{\sim} p_{\text{true}}(x)$$

- maximum meaningful **equivalent number of events**

$$M \left[D_{\text{true}}^{n_{\text{gen}}}, p_{\text{true}} \right] \Big|_{n_{\text{gen}}=n_{\text{equiv}}} \stackrel{!}{=} \lim_{n_{\text{gen}} \rightarrow \infty} M \left[D_{\text{gen}}^{n_{\text{gen}}}, p_{\text{true}} \right]$$

- compare two densities with two uncertainties

$$M = \sigma_{\text{stat}}^2 + \sigma_{\text{model}}^2 = \begin{cases} \frac{a}{n_{\text{gen}}} & n_{\text{gen}} \ll n_{\text{train}} \\ \frac{b}{n_{\text{train}}} & n_{\text{gen}} \gg n_{\text{train}} \end{cases} \quad \text{transition} \quad G = \frac{n_{\text{equiv}}}{n_{\text{train}}} = \frac{a}{b}$$



Generative amplification

How many events can I sample from a network trained on n events?

- learned phase space density

$$D_{\text{true}}^{n_{\text{train}}} \sim p_{\text{true}} \approx p_{\text{gen}}(x) \quad \Rightarrow \quad p_{\text{gen}}(x) \stackrel{?}{\sim} p_{\text{true}}(x)$$

- maximum meaningful **equivalent number of events**

$$M \left[D_{\text{true}}^{n_{\text{gen}}}, p_{\text{true}} \right] \Big|_{n_{\text{gen}}=n_{\text{equiv}}} \stackrel{!}{=} \lim_{n_{\text{gen}} \rightarrow \infty} M \left[D_{\text{gen}}^{n_{\text{gen}}}, p_{\text{true}} \right]$$

- local integral estimate M_I [alternative: Kolmogorov-Smirnov distance]

$$I(p_{\text{true}}) = \int_V dx \, p_{\text{true}}(x) \quad \text{vs} \quad \bar{I}(D_{\text{gen}}^{n_{\text{gen}}}) = \frac{1}{n} \sum_x \mathbf{1}_{x \in V}$$

$$\begin{aligned} \Rightarrow M_I \left[D_{\text{gen}}^{n_{\text{gen}}}, p_{\text{true}} \right] &= \left\langle \left[\bar{I}(D_{\text{gen}}^{n_{\text{gen}}}) - I(p_{\text{true}}) \right]^2 \right\rangle \\ &= \begin{cases} \sigma_{\text{stat}}^2(n_{\text{gen}}) & p_{\text{gen}} = p_{\text{true}} \\ \sigma_{\text{stat}}^2(n_{\text{gen}}) + \sigma_{\text{model}}^2(p_{\text{gen}}, p_{\text{true}}) & p_{\text{gen}} \neq p_{\text{true}} \end{cases} \end{aligned}$$

- binomial distribution to extrapolate

$$M_I \left[D_{\text{true}}^{n_{\text{equiv}}}, p_{\text{true}} \right] = \sigma_{\text{stat}}^2(n_{\text{equiv}}) \simeq \frac{\bar{I}(D_{\text{true}}^{n_{\text{train}}})[1 - \bar{I}(D_{\text{true}}^{n_{\text{train}}})]}{n_{\text{equiv}}}$$

$$\lim_{n_{\text{gen}} \rightarrow \infty} M_I \left[D_{\text{gen}}^{n_{\text{gen}}}, p_{\text{true}} \right] = \sigma_{\text{model}}^2(p_{\text{gen}}, p_{\text{true}})$$

→ **Needing model uncertainty, eventually BNN**



Extrapolating ISR

Universal QCD jet radiation [Z + 1...8 jets]

- from n to $n + 1$ jets

$$R_{(n+1)/n} = \frac{\sigma_{n+1}}{\sigma_n} \quad \text{and} \quad P(n) = \frac{\sigma_n}{\sigma_{\text{tot}}} \quad \text{with} \quad \sigma_{\text{tot}} = \sum_{n=0}^{\infty} \sigma_n .$$

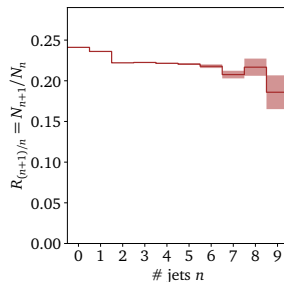
- large scale drop: Poisson scaling

$$R_{(n+1)/n} = \frac{\bar{n}}{n+1} \quad \Leftrightarrow \quad P(n) = \frac{\bar{n}^n e^{-\bar{n}}}{n!} .$$

- democratic scales: staircase scaling

$$R_{(n+1)/n} = e^{-b} = 1 - \tilde{\Delta}_g(Q^2) \equiv P(n+1|n)$$

→ Universal pattern learnable?



Extrapolating ISR

Universal QCD jet radiation [Z + 1...8 jets]

- democratic scales: staircase scaling

$$R_{(n+1)/n} = e^{-b} = 1 - \tilde{\Delta}_g(Q^2) \equiv P(n+1|n)$$

→ Universal pattern learnable?

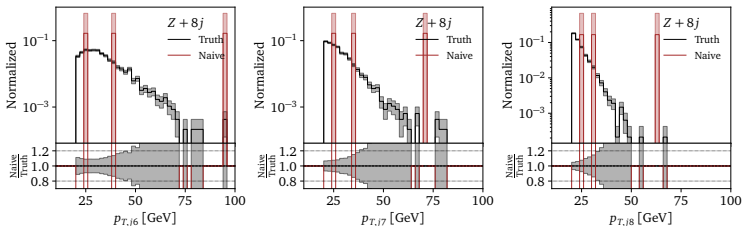
Autoregressive transformer

- factorized probability and loss function

$$p(x_i | x_{1:i-1}) = p_{\text{kin}}(x_i | x_{1:i-1}) p_{\text{split}}(x_{1:i-1})$$

$$p(x_{1:n}) = \left[\prod_{i=1}^n p_{\text{kin}}(x_i | x_{1:i-1}) \right] \left[\prod_{i=1}^n p_{\text{split}}(x_{1:i-1}) \right] [1 - p_{\text{split}}(x_{1:n})],$$

- naive extrapolation, train up to 6 jets, generate 7 and 8



Extrapolating ISR

Universal QCD jet radiation [Z + 1...8 jets]

- democratic scales: staircase scaling

$$R_{(n+1)/n} = e^{-b} = 1 - \tilde{\Delta}_g(Q^2) \equiv P(n+1|n)$$

→ Universal pattern learnable?

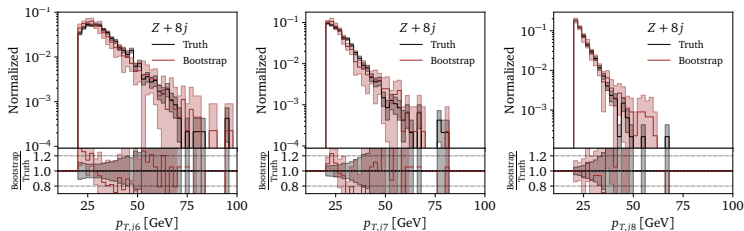
Autoregressive transformer

- factorized probability and loss function

$$p(x_i | x_{1:i-1}) = p_{\text{kin}}(x_i | x_{1:i-1}) p_{\text{split}}(x_{1:i-1})$$

$$p(x_{1:n}) = \left[\prod_{i=1}^n p_{\text{kin}}(x_i | x_{1:i-1}) \right] \left[\prod_{i=1}^n p_{\text{split}}(x_{1:i-1}) \right] [1 - p_{\text{split}}(x_{1:n})],$$

- bootstrapped extrapolation, train up to 6 jets, generate 7 and 8



Extrapolating ISR

Universal QCD jet radiation $[Z + 1 \dots 8 \text{ jets}]$

- democratic scales: staircase scaling

$$R_{(n+1)/n} = e^{-b} = 1 - \tilde{\Delta}_g(Q^2) \equiv P(n+1|n)$$

→ Universal pattern learnable?

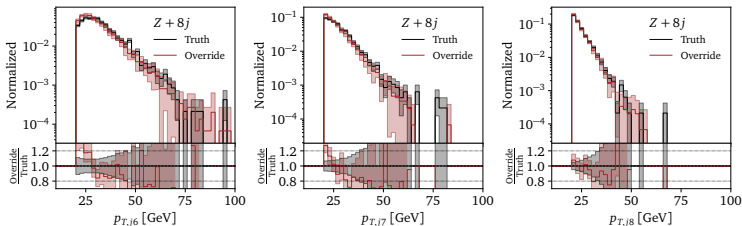
Autoregressive transformer

- factorized probability and loss function

$$p(x_i | x_{1:i-1}) = p_{\text{kin}}(x_i | x_{1:i-1}) p_{\text{split}}(x_{1:i-1})$$

$$p(x_{1:n}) = \left[\prod_{i=1}^n p_{\text{kin}}(x_i | x_{1:i-1}) \right] \left[\prod_{i=1}^n p_{\text{split}}(x_{1:i-1}) \right] [1 - p_{\text{split}}(x_{1:n})],$$

→ full extrapolation with right latent representation



AI for fundamental physics

Develop AI for the best science

- 1 just another tool for a numerical field
- 2 transformative new language
 - many applications in LHC theory

MadGraph7

MLhad

NNLO

Simulation-based inference

Unfolding

SFitter global analyses

- Make complexity our friend
- Remember we still do LHC theory

2211.01421v2 [hep-ph] 27 Mar 2024

Modern Machine Learning for LHC Physicists

Tilman Plehn^a, Anja Butter^{a,b}, Barry Dillon^a,
Theo Heimgartner^c, Claudius Krause^c, and Ramon Winterhalder^d

^a Institut für Theoretische Physik, Universität Heidelberg, Germany

^b LPNHE, Sorbonne Université, Université Paris Cité, CNRS/IN2P3, Paris, France

^c HEPHY, Austrian Academy of Sciences, Vienna, Austria

^d CP3, Université catholique de Louvain, Louvain-la-Neuve, Belgium

March 19, 2024

Abstract

Modern machine learning is transforming particle physics fast, bullying its way into our numerical tool box. For young researchers it is crucial to stay on top of this development, which means applying cutting-edge methods and tools to the full range of LHC physics problems. These lecture notes lead students with basic knowledge of particle physics and significant enthusiasm for machine learning to relevant applications. They start with an LHC-specific motivation and a non-standard introduction to neural networks and then cover classification, unsupervised classification, generative networks, and inverse problems. Two themes defining much of the discussion are well-defined loss functions and uncertainty-aware networks. As part of the applications, the notes include some aspects of theoretical LHC physics. All examples are chosen from particle physics publications of the last few years.¹

