Machine Learning Basics

Tilman Plehn

Universität Heidelberg

PHYSTAT School

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

$\rightarrow$ θ data representation

Probabilities over phase space

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

LHC representations

- accuracy [checking bias]
- precision [calibrated uncertainties, coverage]
- structure [symmetries etc]

$\rightarrow$ 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)

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

$\rightarrow$ maximize tractable likelihood instead [assume Gaussian]

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

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

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)

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

$\rightarrow$ maximize tractable likelihood instead [assume Gaussian]

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

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

Learned local uncertainty

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

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

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

$\rightarrow$ 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)

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

$\to$ maximize tractable likelihood instead [assume Gaussian]

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

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

Negative log-likelihood minimization

- target function preprocessing

$$f_j \to \log f_j \qquad \text{or} \qquad f_j \to f_j - \langle f_j \rangle \qquad \text{or} \qquad f_i \to \frac{f_i}{\langle f_i \rangle} \cdots$$

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

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

$\to$ 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

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

- 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 $\rho(\theta, t) = \pi(\theta)$

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

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

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

- 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

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

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

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

- 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

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

$\rightarrow$ 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

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

- 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]

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

$\rightarrow$ 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]

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

$\rightarrow$ 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]

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

$\rightarrow$ Two-term loss: likelihood + prior

2- Bayesian networks

Statistical network evaluation

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

$$\langle A \rangle = \int dA d\theta \; A \; p(A|\theta) \; q(\theta)$$

$$\equiv \int d\theta \; q(\theta) \overline{A}(\theta) \qquad \text{with} \qquad \overline{A}(\theta) = \int dA \; A \; p(A|\theta)$$

- corresponding variance

$$\sigma^2_{\text{tot}} = \int dA d\theta \;\; (A - \langle A \rangle)^2 \;\; p(A|\theta) \; q(\theta)$$

$$= \int d\theta \; q(\theta) \left[\overline{A^2}(\theta) - \overline{A}(\theta)^2 + \Big(\overline{A}(\theta) - \langle A \rangle \Big)^2 \right] \equiv \sigma^2_{\text{syst}} + \sigma^2_{\text{stat}}$$

Two uncertainties

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

$$\sigma^2_{\text{stat}} = \int d\theta \; q(\theta) \left[\overline{A}(\theta) - \langle A \rangle \right]^2$$

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

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

- (ir)-reducible errors in ML language

$\to$ Systematics dominant for LHC

2- Bayesian vs deterministic networks

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

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

$\rightarrow$ 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

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

$\to$ Likelihood with L2-regularization

Dropout

- Bernoulli weights

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

- 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...

$\to$ 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

$$\begin{aligned}
\langle A\rangle &= \int dA\; A\; p(A) \\
&= \int d\bar{A} d\sigma^2\; \bar{A}\; q_m(\bar{A},\sigma^2) = \gamma \\
\sigma^2_{\text{syst}} &= \int d\bar{A} d\sigma^2\; \sigma^2\; q_m(\bar{A},\sigma^2) = \frac{\beta}{\alpha-1} \\
\sigma^2_{\text{stat}} &= \int d\bar{A} d\sigma^2 \left[\bar{A} - \langle A\rangle\right]^2\; q_m(\bar{A},\sigma^2) = \frac{\beta}{v(\alpha-1)}
\end{aligned}$$

- 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

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

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

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

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

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

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

$$p(A) = \int d\overline{A} d\sigma^2 \mathcal{N}(A|\overline{A},\sigma^2)\; q_m(\overline{A},\sigma^2)$$

$$= \text{St}\left(A\middle|\gamma, \frac{\beta(1+v)}{v\alpha}, 2\alpha\right)$$

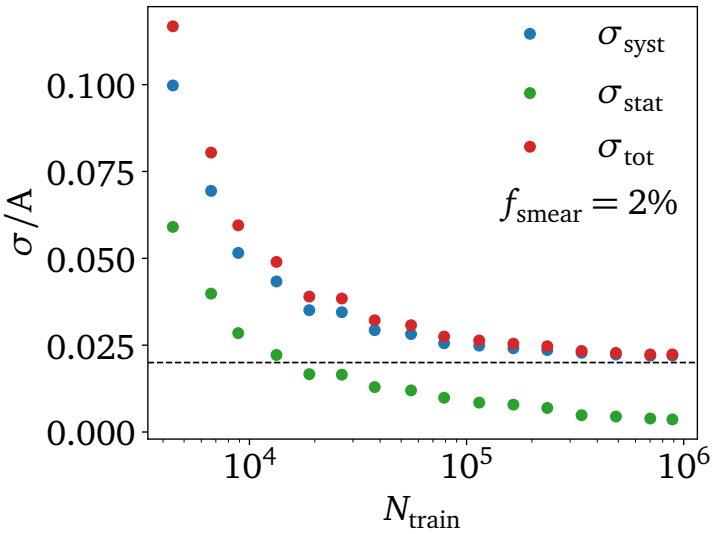
- generalizable to Gaussian mixture model

$\rightarrow$ Another alternative for learned uncertainty

Amplitudes with calibrated uncertainties

Amplitude regression

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

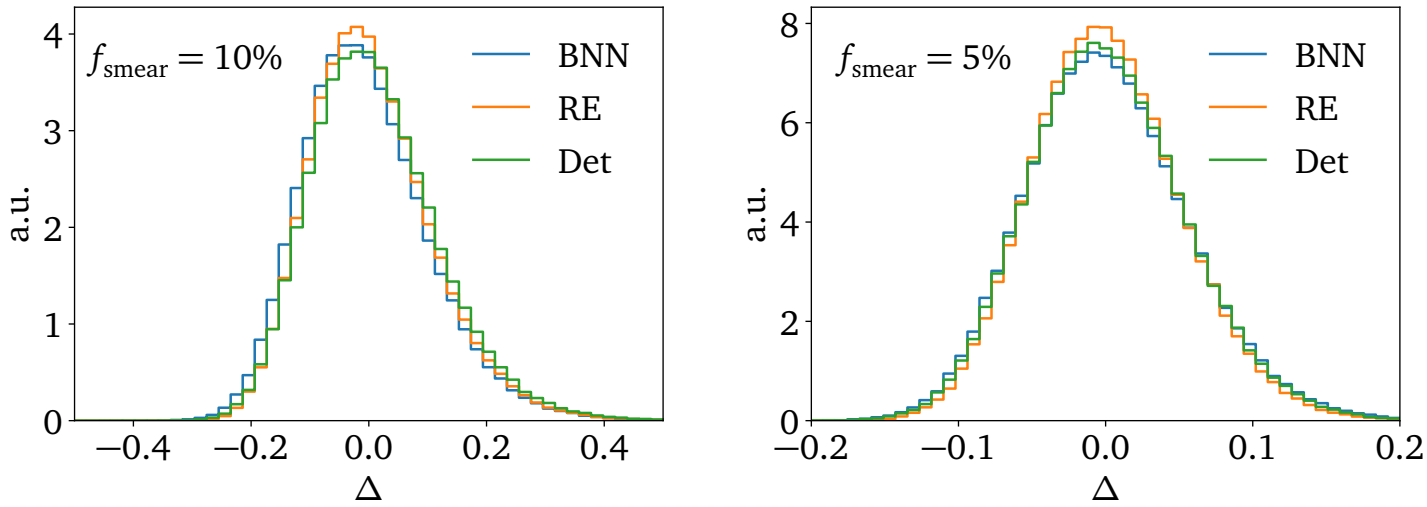


Amplitudes with calibrated uncertainties

Amplitude regression

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

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



Amplitudes with calibrated uncertainties

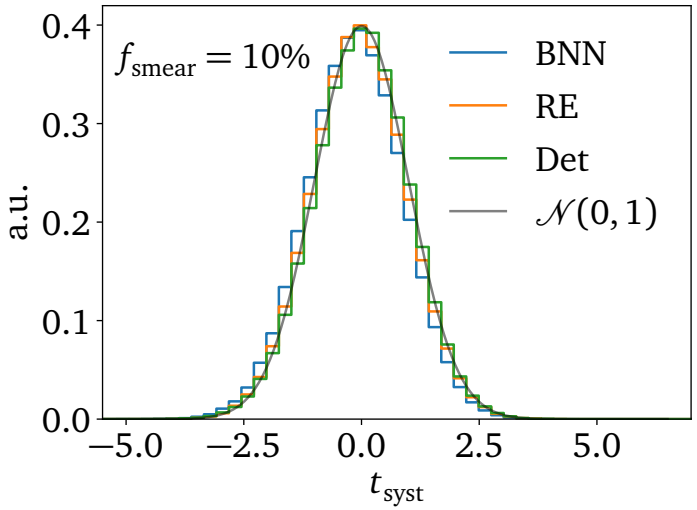
Amplitude regression

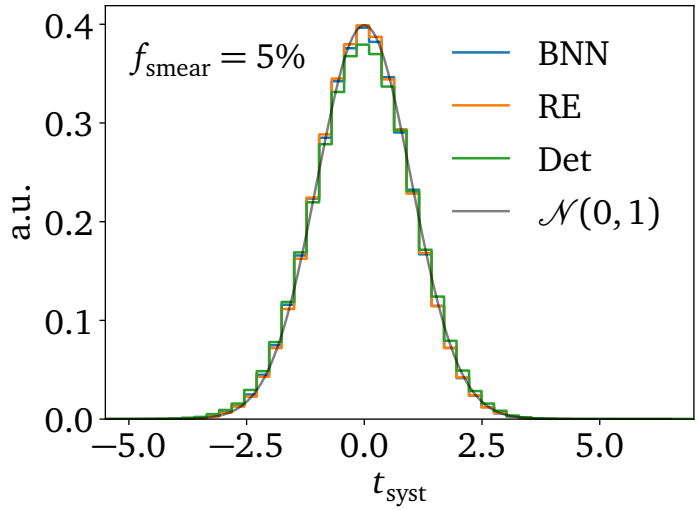
- loop amplitude over phase space $gg \to \gamma\gamma g(g)$
- systematics: artificial noise
- statistics plateau
- accuracy over phase space

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

- 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 $gg \to \gamma\gamma g(g)$
- 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

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

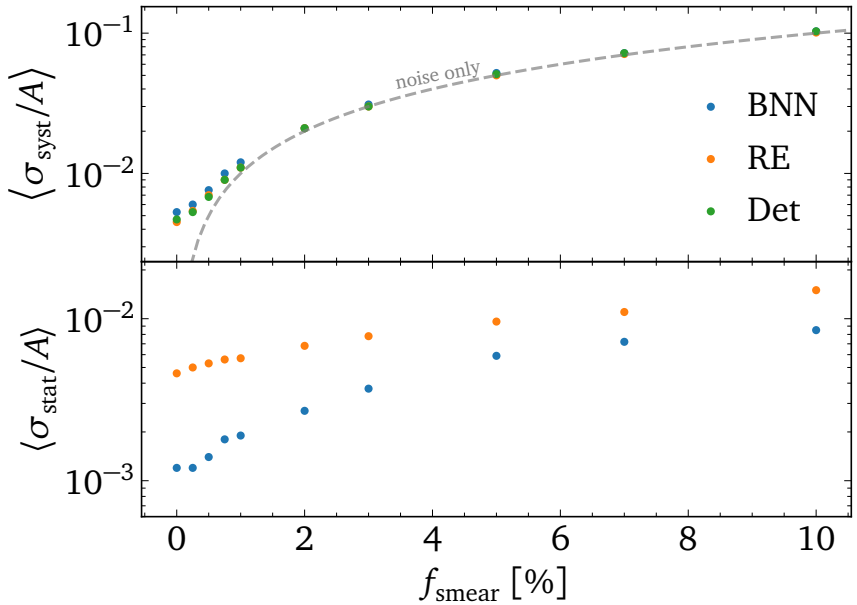
Towards zero noise

- scaling

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

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

$\to$ Limiting factor??



Amplitudes with calibrated uncertainties

Amplitude regression

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

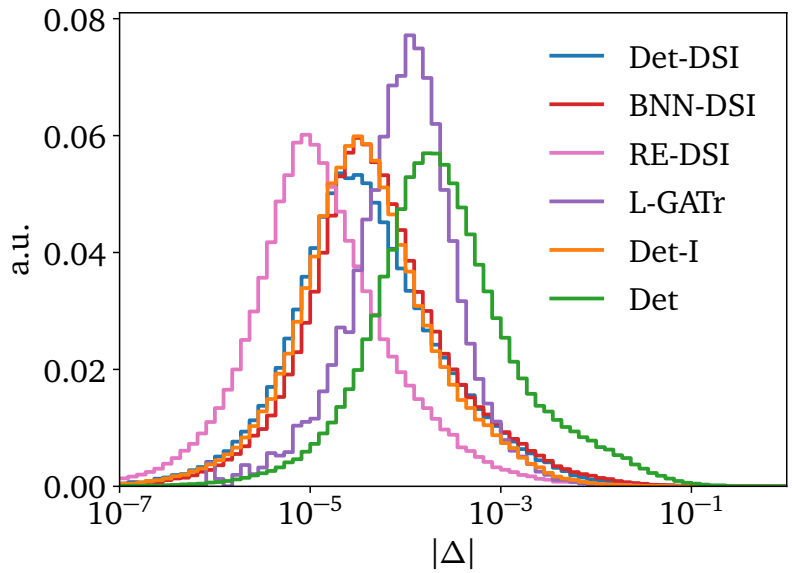
$$\Delta(x) = \frac{A_{\text{NN}}(x) - A_{\text{true}}(x)}{A_{\text{true}}(x)}$$

- pull over phase space

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

Data representation

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



Amplitudes with calibrated uncertainties

Amplitude regression

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

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

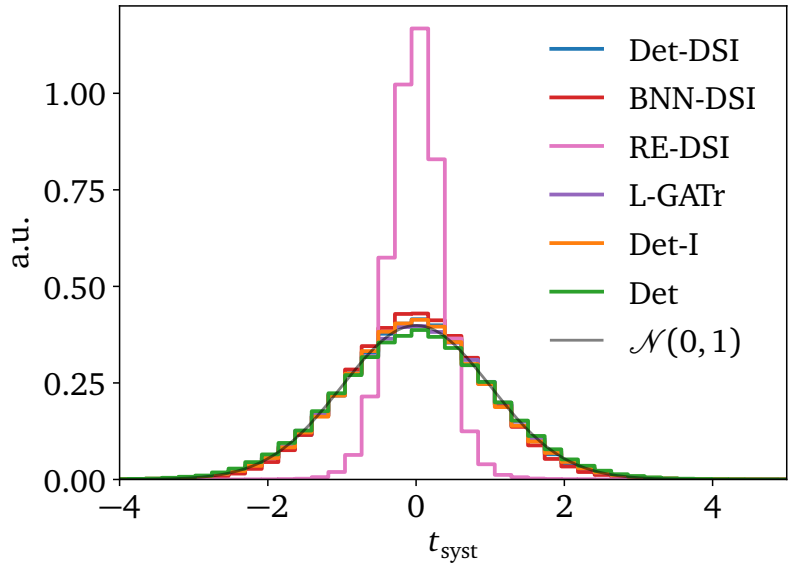
- pull over phase space

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

Data representation

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

$\to$ Calibrated systematics



Amplitudes with calibrated uncertainties

Amplitude regression

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

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

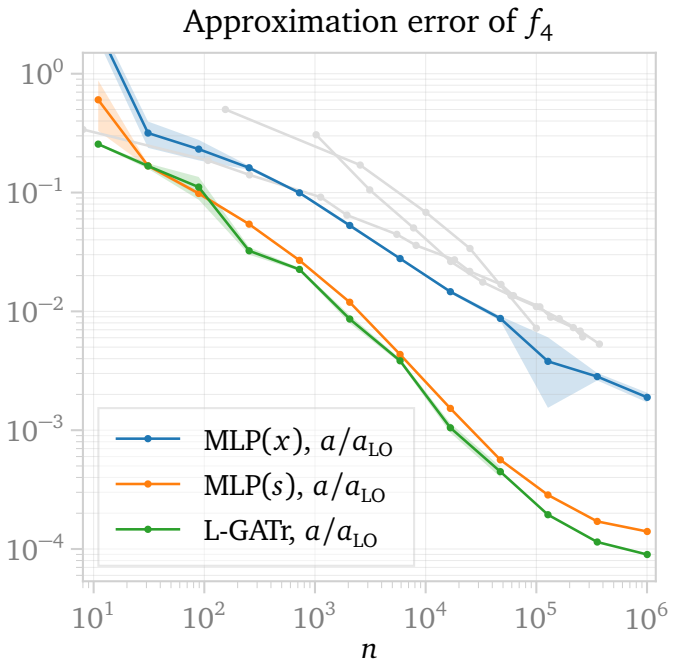
- pull over phase space

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

Data representation

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

$\to$ On to real loop integrals



Phase space gaps

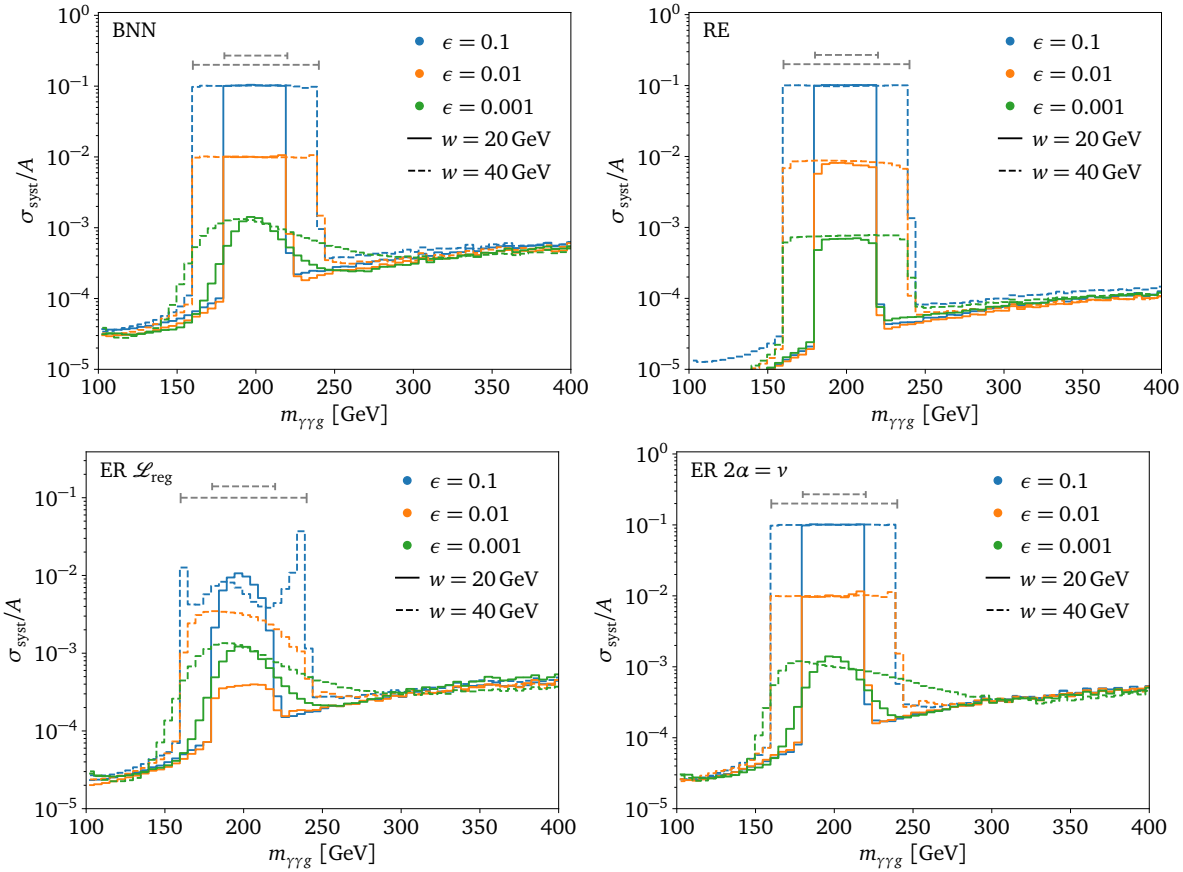
Noisy amplitude gap

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

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

- 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 $x \to z$
- 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

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

$\to$ Permutation-eqivariant 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 $x \to z$
- 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

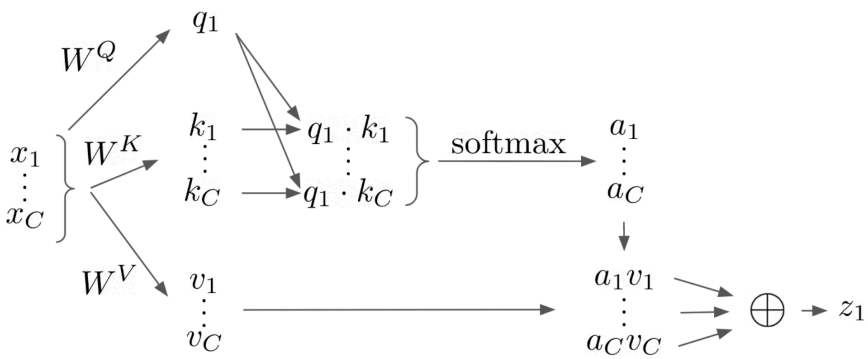
$$q = \sum_j (q \cdot k_j)\, v_j \quad \text{with} \quad k_j = \frac{v_j}{v^2}$$

$\to$ Permutation-eqivariant self-attention representation

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

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 $\to$ scalars, vectors, etc?
- geometric product extending vector space

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

- 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]

$$x = x^S\, 1 + x^V_\mu\, \gamma^\mu + x^B_{\mu\nu}\, \sigma^{\mu\nu} + x^A_\mu\, \gamma^\mu \gamma^5 + x^P\, \gamma^5 \qquad \text{with} \qquad \begin{pmatrix} x^S \\ x^V_\mu \\ x^B_{\mu\nu} \\ x^A_\mu \\ x^P \end{pmatrix} \in \mathbb{R}^{16}$$

- example input (PID, p_μ)

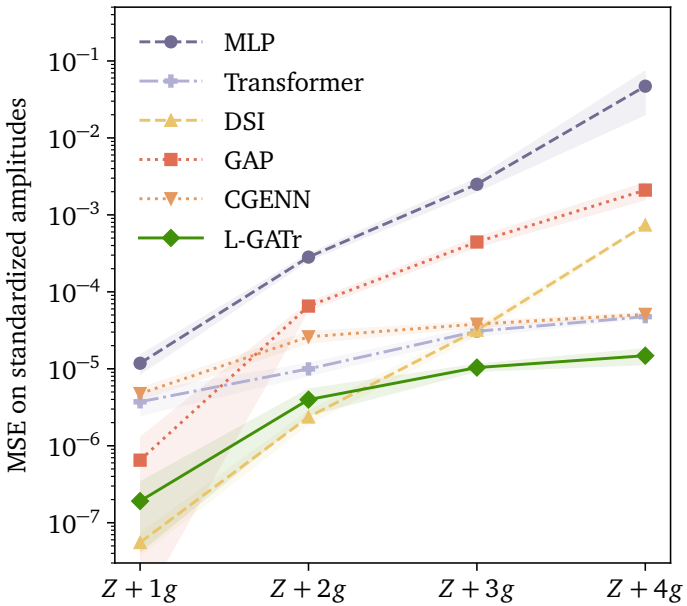
$$x^S = \text{PID} \qquad x^V_\mu = p_\mu \qquad x^T_{\mu\nu} = x^A_\mu = x^P = 0$$

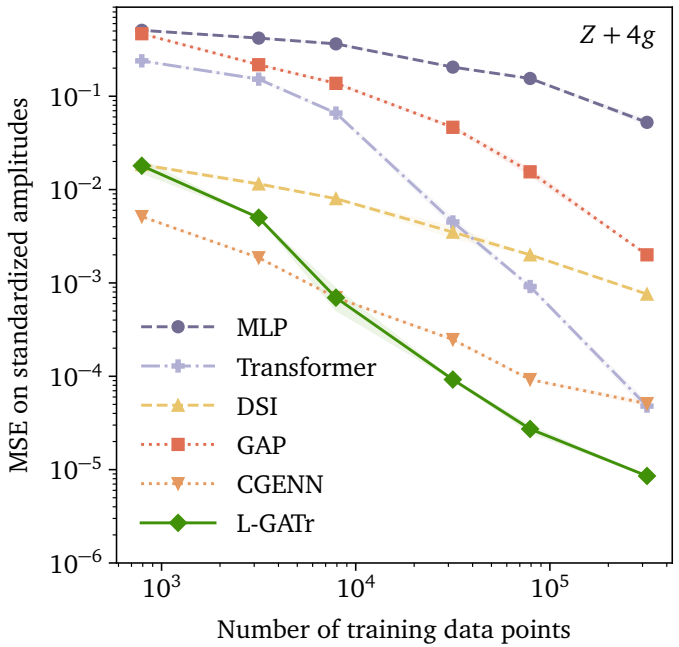
$\to$ Organize network layers by grade: L-GATr

L-GATr amplitudes

Performance is all you need

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



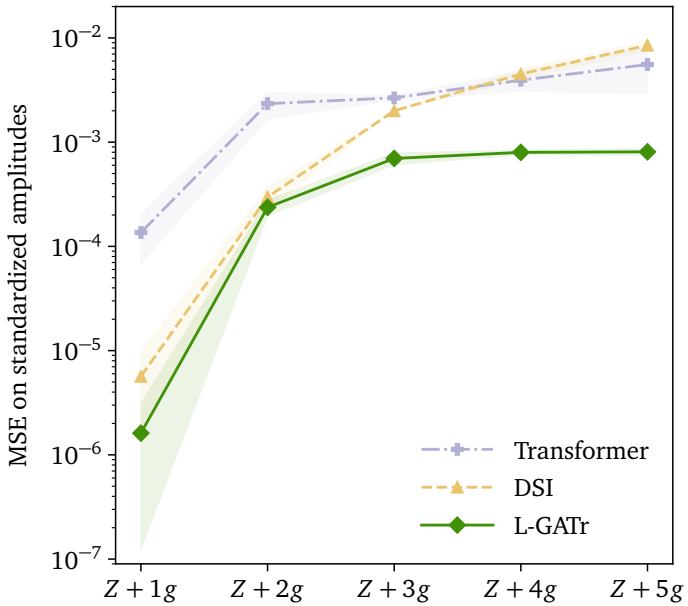


L-GATr amplitudes

Performance is all you need

- LO transition amplitudes $q\bar{q} \to Z + 1\ldots4\ g$
- amplitude regression: cost scaling with dimensionality
  $\to$ low-dimensional representation better
- performance scaling with multiplicity and training data
  $\to$ winning transformers
- more scaling with less data
  $\to$ equivariance with additional advantage

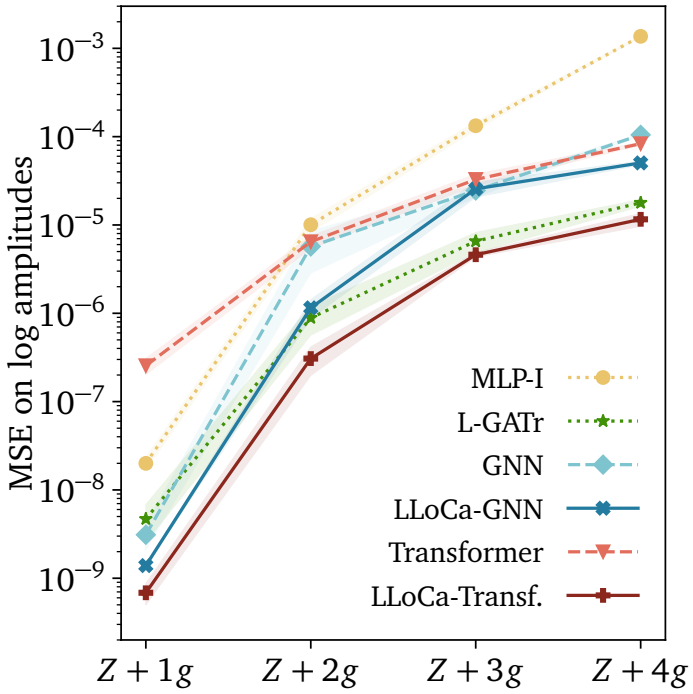
$\to$ Equvariant transformers current winner

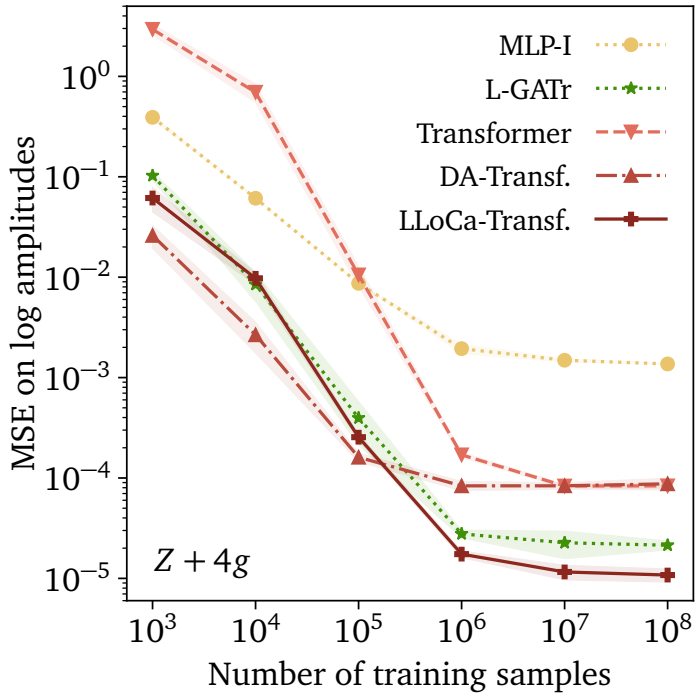


LLoCa vs L-GATr amplitudes

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
  $\rightarrow$ L-GATs and LLoCa comparable



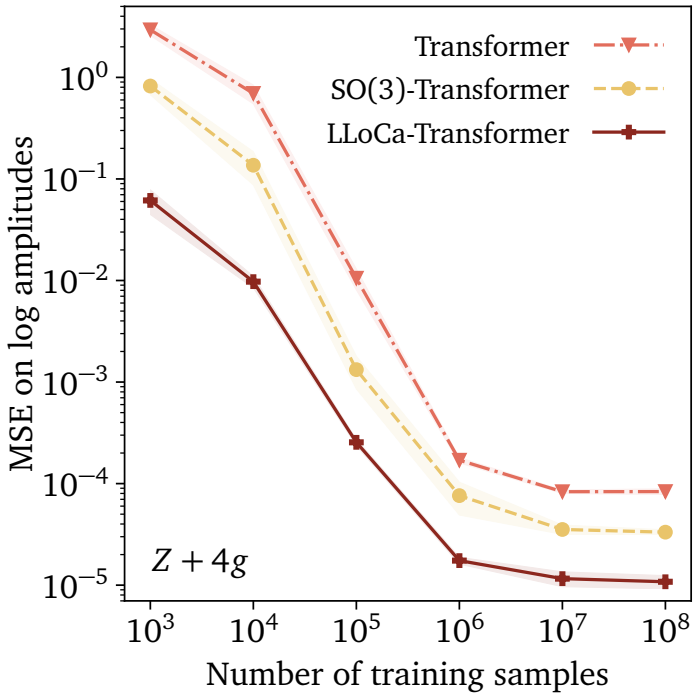


LLoCa vs L-GATr amplitudes

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
  $\to$ L-GATs and LLoCa comparable
- broken symmetries: rotations
  $\to$ intermediate symmetries means intermediate performance

$\to$ Advantage of equivariance for all implementations



Invariant latent representation

Symmetry from simulations $f_\theta(S(x)) = f_\theta(x)$

- self-supervised training of latent representation
- given training objects x_i
  generated objects $x'_i = S(x_i)$ [augmentation]

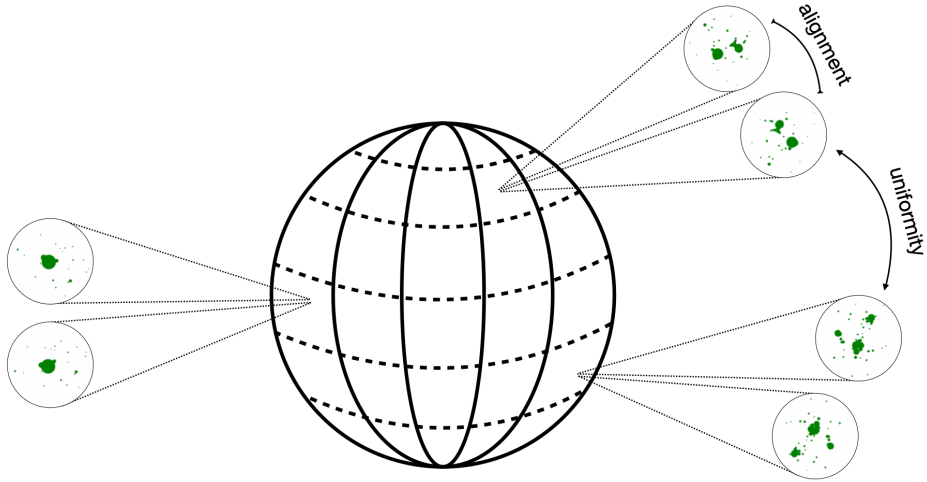
  positive pairs $\{(x_i, x'_i)\}$

  negative pairs $\{(x_i, x_j)\} \cup \{(x_i, x'_j)\}$ for $i \neq j$
- compact latent representation for finite distance measure

$$f_\theta(x_i) = \frac{z_i}{|z_i|} \quad \Rightarrow \quad s(z_i, z_j) = \frac{z_i \cdot z_j}{|z_i||z_j|} \in [-1, 1]$$

- contrasive alignment-uniformity loss

$$\mathcal{L}_{\text{CLR}} = -\sum_i \frac{s(z_i, z'_i)}{\tau} + \sum_i \log \sum_{i \neq i} \left[e^{s(z_i, z_j)/\tau} + e^{s(z_i, z'_j)/\tau} \right]$$



Invariant latent representation

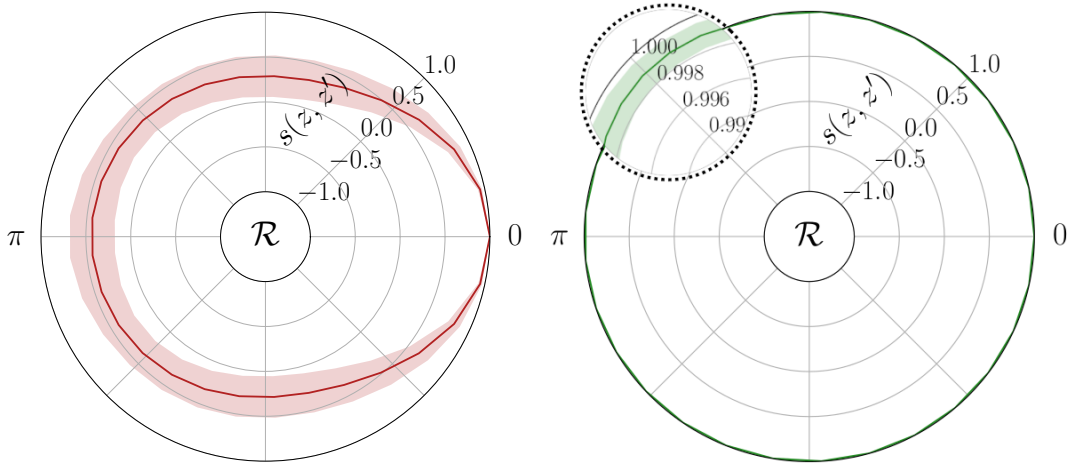
Symmetry from simulations $f_\theta(S(x)) = f_\theta(x)$

- self-supervised training of latent representation
- given training objects x_i
  generated objects $x'_i = S(x_i)$ [augmentation]
- contrasive alignment-uniformity loss

$$\mathcal{L}_{\text{CLR}} = -\sum_i \frac{s(z_i, z'_i)}{\tau} + \sum_i \log \sum_{j\neq i} \left[e^{s(z_i,z_j)/\tau} + e^{s(z_i,z'_j)/\tau} \right]$$

QCD jets

- translations
  rotations around jet axis
  soft-collinear radiation
- test of rotation-invariant representation



Invariant latent representation

Symmetry from simulations $f_\theta(S(x)) = f_\theta(x)$

- self-supervised training of latent representation
- given training objects x_i
  generated objects $x'_i = S(x_i)$ [augmentation]
- contrasive alignment-uniformity loss

$$\mathcal{L}_{\text{CLR}} = -\sum_i \frac{s(z_i, z'_i)}{\tau} + \sum_i \log \sum_{j\neq i} \left[e^{s(z_i,z_j)/\tau} + e^{s(z_i,z'_j)/\tau} \right]$$

QCD jets

- translations
  rotations around jet axis
  soft-collinear radiation
- test of rotation-invariant representation
- test with linear-layer classification

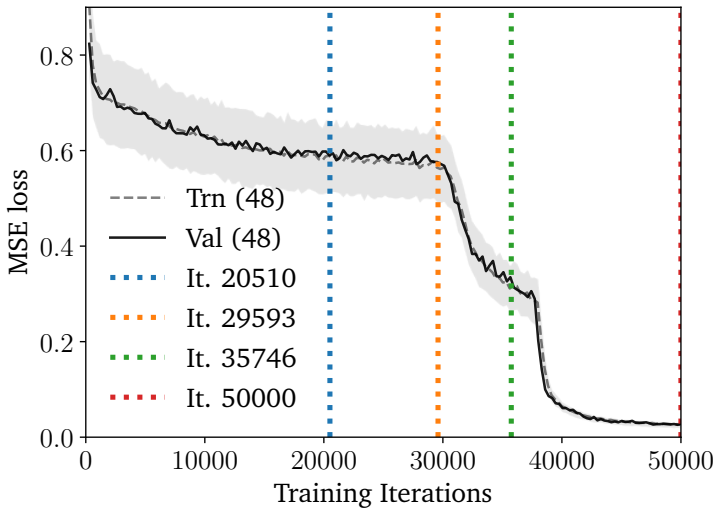
$\rightarrow$ Basis of foundation model training

Augmentation	$\epsilon^{-1}(\epsilon_s = 0.5)$	AUC
none	15	0.905
translations	19	0.916
rotations	21	0.930
soft+collinear	89	0.970
combined	181	0.979

Symmetry-aware representations?

Amplitude ratio regression

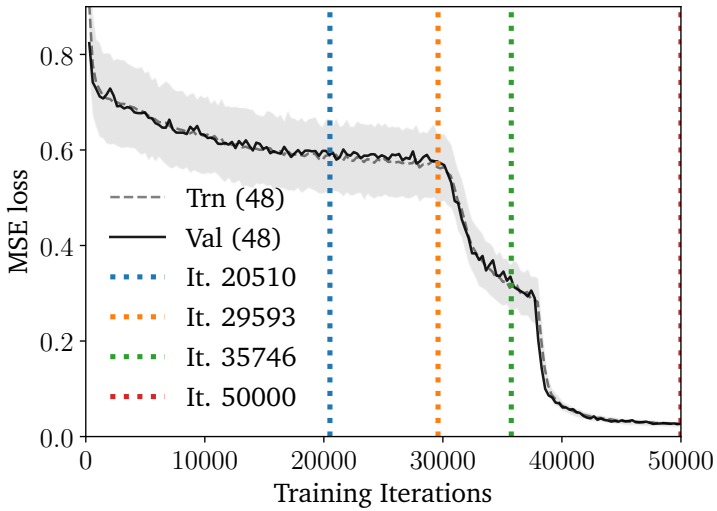
- full vs. leading color for $gg \to 4g$
- standard transformer training [not for MLP, GNN, L-GATr]

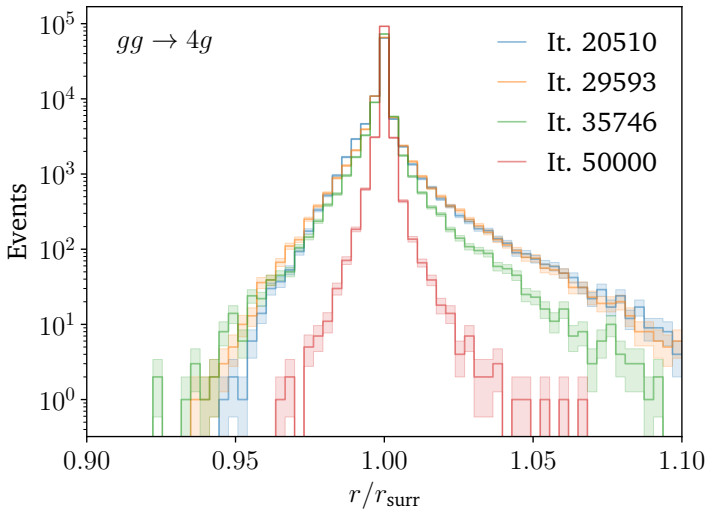


Symmetry-aware representations?

Amplitude ratio regression

- full vs. leading color for $gg \to 4g$
- standard transformer training [not for MLP, GNN, L-GATr]
- related to performance gain
- What is happening?





Symmetry-aware representations?

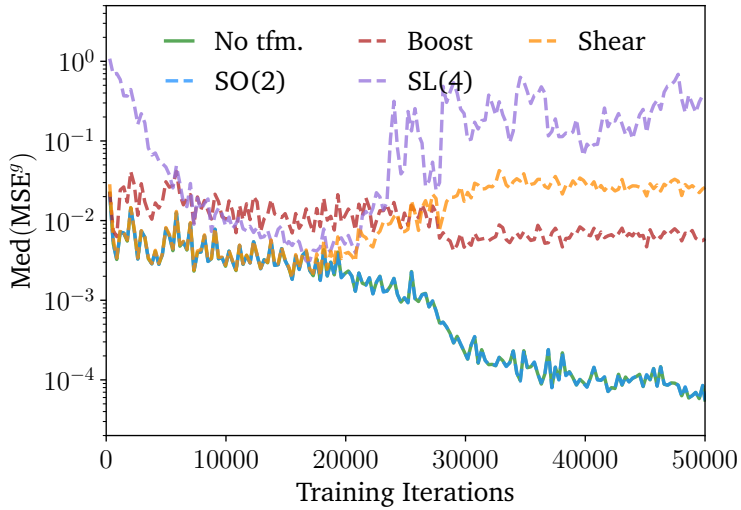
Amplitude ratio regression

- full vs. leading color for $gg \to 4g$
- standard transformer training [not for MLP, GNN, L-GATr]
- related to performance gain
- What is happening?

Symmetries?

- evaluate MSE on transformed data

  wrong: 4D rotation SL(4), $y - z$-shear
  right: Lorentz boosts, SO(2) rotations
- initially learning general structures
  then penalizing wrong symmetries
  Lorentz symmetries ...
  SO(2) always good symmetry
- What can we do with it?



Symmetry encoding

Systematize symmetry-aware layers

- point cloud with graph representation: nodes x_i, edges e_{ij}
- convolutional networks [neighborhood $j \in N$]
  translation symmetry, weight sharing
- in terms of ReLu layer with learned F_θ
  edges c_{ij} to learned node features ϕ_θ

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} c_{ij} \phi_\theta(x_j) \right]$$

$\rightarrow$ 'attention is all you need' winning motto of orthogonal bases?

Symmetry encoding

Systematize symmetry-aware layers

- point cloud with graph representation: nodes x_i, edges e_{ij}
- convolutional networks [neighborhood $j \in N$]
  translation symmetry, weight sharing
- in terms of ReLu layer with learned F_θ
  edges c_{ij} to learned node features ϕ_θ

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} c_{ij} \phi_\theta(x_j) \right]$$

- self attention in transformers
  general edge definition $a(x_i, x_j, e_{ij})$

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} a(x_i, x_j, e_{ij}) \phi_\theta(x_j) \right]$$

$\rightarrow$ 'attention is all you need' winning motto of orthogonal bases?

Symmetry encoding

Systematize symmetry-aware layers

- point cloud with graph representation: nodes x_i, edges e_{ij}
- convolutional networks [neighborhood $j \in N$]
  translation symmetry, weight sharing
- in terms of ReLu layer with learned F_θ
  edges c_{ij} to learned node features ϕ_θ

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} c_{ij} \phi_\theta(x_j) \right]$$

- self attention in transformers
  general edge definition $a(x_i, x_j, e_{ij})$

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} a(x_i, x_j, e_{ij}) \phi_\theta(x_j) \right]$$

- message passing graph networks
  combined node-edge function

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} \phi_\theta(x_i, x_j, e_{ij}) \right]$$

$\rightarrow$ 'attention is all you need' winning motto of orthogonal bases?

Symmetry encoding

Systematize symmetry-aware layers

- point cloud with graph representation: nodes x_i, edges e_{ij}
- convolutional networks [neighborhood $j \in N$]
  translation symmetry, weight sharing
- in terms of ReLu layer with learned F_θ
  edges c_{ij} to learned node features ϕ_θ

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} c_{ij} \phi_\theta(x_j) \right]$$

- self attention in transformers
  general edge definition $a(x_i, x_j, e_{ij})$

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} a(x_i, x_j, e_{ij}) \phi_\theta(x_j) \right]$$

- message passing graph networks
  combined node-edge function

$$x_i' = \text{ReLu } F_\theta \left[x_i, \square_{j \in N} \phi_\theta(x_i, x_j, e_{ij}) \right]$$

- simplified deep sets
  assuming the minimal mathematically needed structure

$$x_i' = \text{ReLu } F_\theta \left[\square_j \phi_\theta(x_j) \right]$$

$\rightarrow$ 'attention is all you need' winning motto of orthogonal bases?

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

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

- perfect training

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

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

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

- 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

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

Neyman-Pearson lemma

- classifier

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

- loss

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

- loss minimum: Neyman-Pearson lemma

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

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

$\to$ Classifying = density ratio estimation

Classification without labels

Classifier training on mixed samples

- signal ratios f_1 and f_2

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

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

- trainable likelihood ratio

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

- relation to signal-background ratio

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

$\rightarrow$ Monotonuous function for working points [fix calibration]

Classification without labels

Classifier training on mixed samples

- signal ratios f_1 and f_2

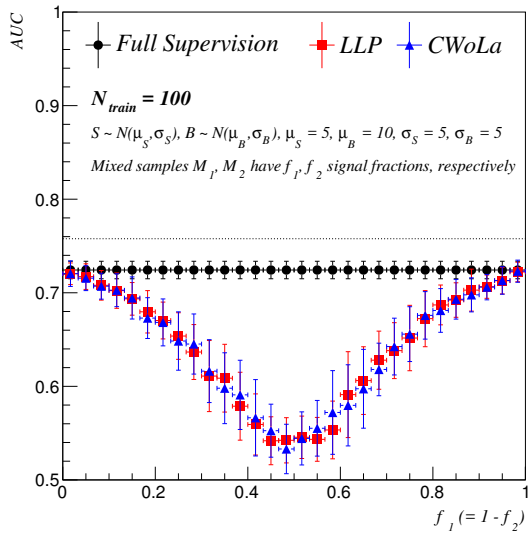
$$\begin{pmatrix} p_1(x) \\ p_2(x) \end{pmatrix} = \begin{pmatrix} f_1 & 1-f_1 \\ f_2 & 1-f_2 \end{pmatrix} \begin{pmatrix} p_S(x) \\ p_B(x) \end{pmatrix}$$

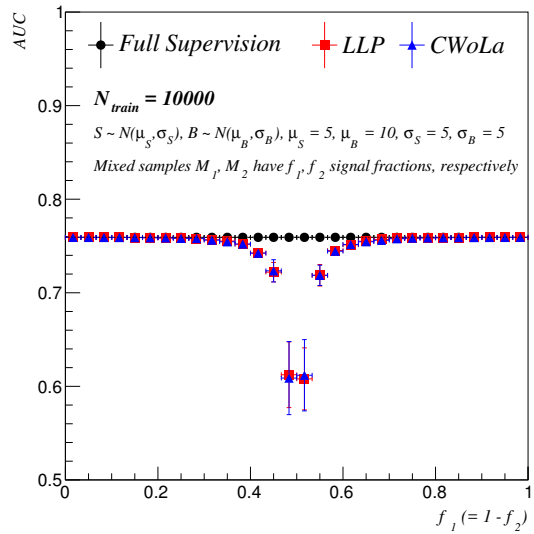
$$\Leftrightarrow \quad \begin{pmatrix} p_S(x) \\ p_B(x) \end{pmatrix} = \frac{1}{f_1 - f_2} \begin{pmatrix} 1-f_2 & f_1-1 \\ -f_2 & f_1 \end{pmatrix} \begin{pmatrix} p_1(x) \\ p_2(x) \end{pmatrix}$$

- relation to signal-background ratio

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

$\rightarrow$ Monotonuous function for working points [fix calibration]

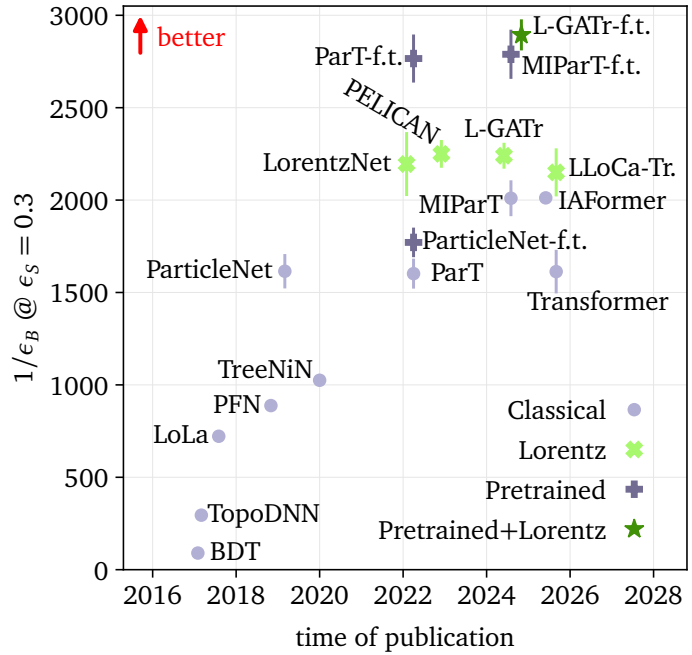




Equivariant jet tagging

Fat jet tagging benchmarks [same bottom line as flavor tagging]

- problem: jet axis breaking Lorentz-symmetry
  1- give jet axis explicitly as additional 4-vector
  2- reduce LLoCa symmetry
- top tagging performance
  $\rightarrow$ equivariance + pre-training leading
  $\rightarrow$ 30-fold background rejection from best BDT



Equivariant jet tagging

Fat jet tagging benchmarks [same bottom line as flavor tagging]

- problem: jet axis breaking Lorentz-symmetry
  1- give jet axis explicitly as additional 4-vector
  2- reduce LLoCa symmetry
- top tagging performance
  $\to$ equivariance + pre-training leading
  $\to$ 30-fold background rejection from best BDT
- multi-class performance
  $\to$ equivariance still leading

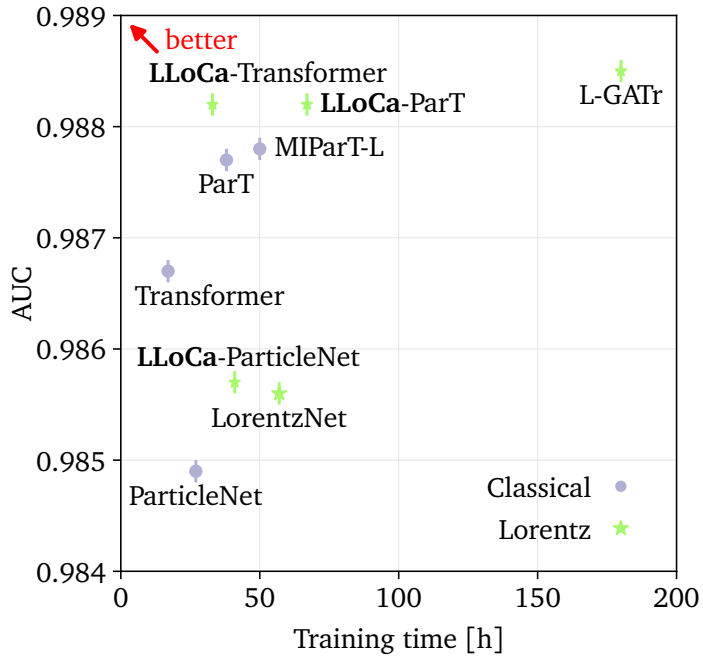
	$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to l\nu q\bar{q}'$	$t \to bq\bar{q}'$	$t \to bl\nu$	$W \to q\bar{q}'$	$Z \to q\bar{q}$
	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{99.5\%}$	$\text{Rej}_{50\%}$	$\text{Rej}_{50\%}$
PFN [72]	2924	841	75	198	265	797	721	189	159
P-CNN [73]	4890	1276	88	474	947	2907	2304	241	204
MIParT-L [68]	10753	4202	123	1927	5450	31250	16807	542	402
LorentzNet [22]	8475	2729	111	1152	3515	13889	10257	400	303
L-GATr [26]	12987	4819	128	2311	6116	47619	20408	588	432
ParticleNet [18]	7634	2475	104	954	3339	10526	11173	347	283
LLoCa-ParticleNet*	7463	2833	105	1072	3155	10753	9302	403	306
ParT [19]	10638	4149	123	1864	5479	32787	15873	543	402
LLoCa-ParT*	11561	4640	125	2037	5900	41667	19231	552	419
Transformer	10753	3333	116	1369	4630	24390	17857	415	334
LLoCa-Transformer*	11628	4651	125	2037	5618	39216	17241	548	410

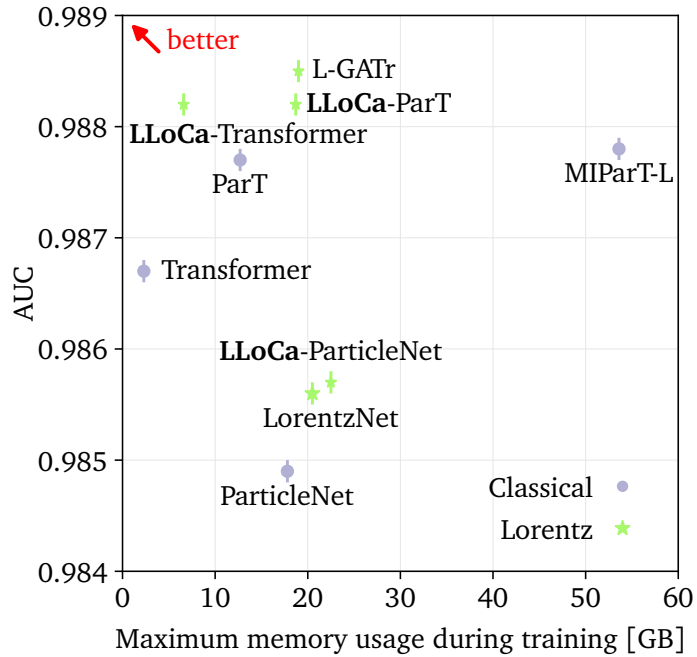
Equivariant jet tagging

Fat jet tagging benchmarks [same bottom line as flavor tagging]

- problem: jet axis breaking Lorentz-symmetry
  1- give jet axis explicitly as additional 4-vector
  2- reduce LLoCa symmetry
- top tagging performance
  $\to$ equivariance + pre-training leading
  $\to$ 30-fold background rejection from best BDT
- multi-class performance
  $\to$ equivariance still leading

$\to$ Efficiency is what you also need...





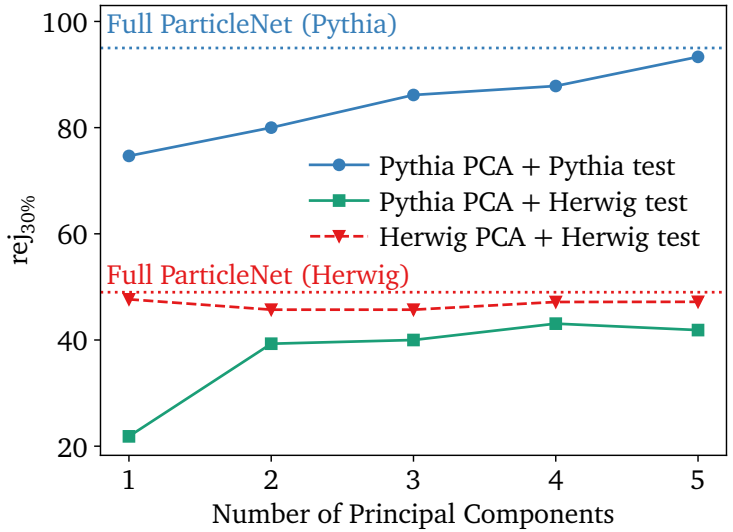
Physics from latent representation

Quarks vs gluons from trained ParticleNet

- sensitive substructure variables

$$n_{\text{pf}} = \sum_i 1 \quad w_{\text{pf}} = \frac{\sum_i p_{T,i} \Delta R_{i,\text{jet}}}{p_{T,\text{jet}}} \quad p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \quad C_\beta = \frac{\sum_{i<j} p_{T,i} p_{T,j} (\Delta R_{ij})^\beta}{\left(\sum_i p_{T,i}\right)^2}$$

- related to max 5 principle components? [linear decorrelation]



Physics from latent representation

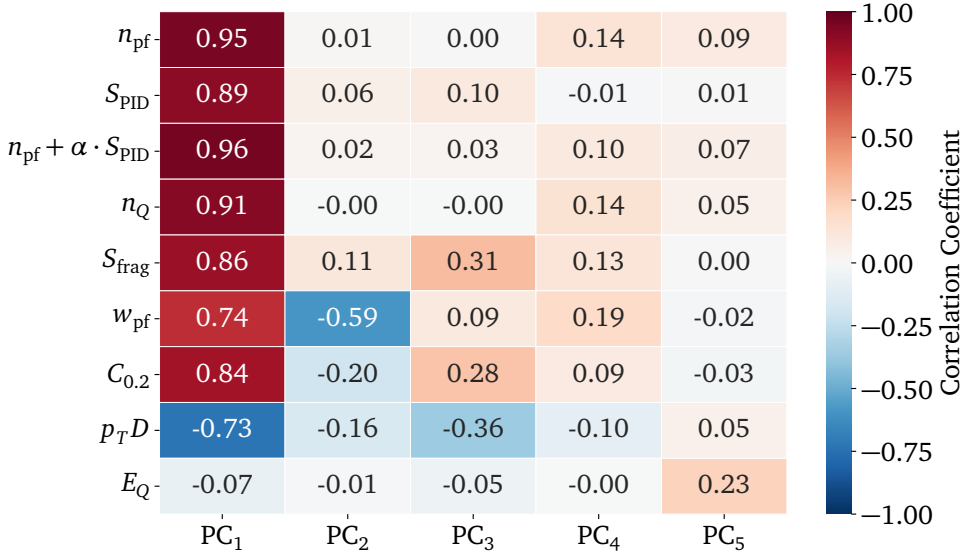
Quarks vs gluons from trained ParticleNet

- sensitive substructure variables

$$n_{\text{pf}} = \sum_i 1 \qquad w_{\text{pf}} = \frac{\sum_i p_{T,i} \Delta R_{i,\text{jet}}}{p_{T,\text{jet}}} \qquad p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \qquad C_\beta = \frac{\sum_{i<j} p_{T,i} p_{T,j} (\Delta R_{ij})^\beta}{\left(\sum_i p_{T,i}\right)^2}$$

- related to max 5 principle components? [linear decorrelation]
- PC_1: constituent number and diversity

$$n_{\text{pf}} + \alpha \cdot S_{\text{PID}} \qquad \text{with} \qquad S_{\text{PID}} = -\sum_{\text{type}j} f_j \log f_j$$



Physics from latent representation

Quarks vs gluons from trained ParticleNet

- sensitive substructure variables

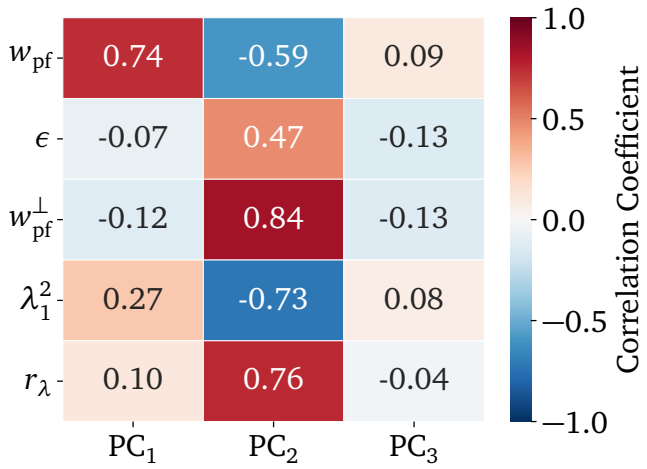
$$n_{\text{pf}} = \sum_i 1 \qquad w_{\text{pf}} = \frac{\sum_i p_{T,i} \Delta R_{i,\text{jet}}}{p_{T,\text{jet}}} \qquad p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \qquad C_\beta = \frac{\sum_{i<j} p_{T,i} p_{T,j} (\Delta R_{ij})^\beta}{\left(\sum_i p_{T,i}\right)^2}$$

- related to max 5 principle components? [linear decorrelation]
- PC_1: constituent number and diversity

$$n_{\text{pf}} + \alpha \cdot S_{\text{PID}} \qquad \text{with} \qquad S_{\text{PID}} = -\sum_{\text{type}j} f_j \log f_j$$

- PC_2: radial energy profile

$$w_{\text{pf}}^\perp = \alpha \cdot n_{\text{pf}} - w_{\text{pf}} \qquad \text{and} \qquad r_\lambda = \frac{\lambda_{0.5}^1}{\lambda_1^2} \qquad \lambda_k^\beta = \sum_i z_i^\beta \Delta R^k$$



Physics from latent representation

Quarks vs gluons from trained ParticleNet

- sensitive substructure variables

$$n_{\text{pf}} = \sum_i 1 \qquad w_{\text{pf}} = \frac{\sum_i p_{T,i} \Delta R_{i,\text{jet}}}{p_{T,\text{jet}}} \qquad p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \qquad C_\beta = \frac{\sum_{i<j} p_{T,i} p_{T,j} (\Delta R_{ij})^\beta}{\left(\sum_i p_{T,i}\right)^2}$$

- related to max 5 principle components? [linear decorrelation]
- PC_1: constituent number and diversity

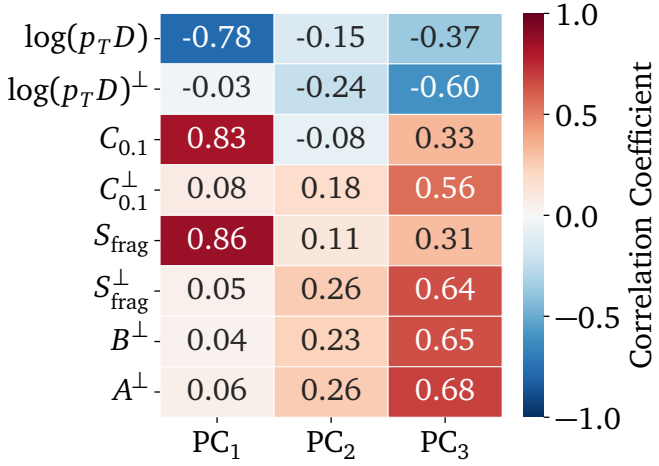
$$n_{\text{pf}} + \alpha \cdot S_{\text{PID}} \qquad \text{with} \qquad S_{\text{PID}} = -\sum_{\text{type}j} f_j \log f_j$$

- PC_2: radial energy profile

$$w_{\text{pf}}^\perp = \alpha \cdot n_{\text{pf}} - w_{\text{pf}} \qquad \text{and} \qquad r_\lambda = \frac{\lambda_{0.5}^1}{\lambda_1^2} \qquad \lambda_k^\beta = \sum_i z_i^\beta \Delta R^k$$

- PC_3: fragmentation and energy dispersion

$$S_{\text{frag}} = -\sum_i z_i \log z_i$$



Physics from latent representation

Quarks vs gluons from trained ParticleNet

- sensitive substructure variables

$$n_{\text{pf}} = \sum_i 1 \quad w_{\text{pf}} = \frac{\sum_i p_{T,i} \Delta R_{i,\text{jet}}}{p_{T,\text{jet}}} \quad p_T D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}} \quad C_\beta = \frac{\sum_{i<j} p_{T,i} p_{T,j} (\Delta R_{ij})^\beta}{\left(\sum_i p_{T,i}\right)^2}$$

- related to max 5 principle components? [linear decorrelation]
- PC_1: constituent number and diversity

$$n_{\text{pf}} + \alpha \cdot S_{\text{PID}} \qquad \text{with} \qquad S_{\text{PID}} = -\sum_{\text{type}j} f_j \log f_j$$

- PC_2: radial energy profile

$$w_{\text{pf}}^\perp = \alpha \cdot n_{\text{pf}} - w_{\text{pf}} \qquad \text{and} \qquad r_\lambda = \frac{\lambda_{0.5}^1}{\lambda_1^2} \qquad \lambda_k^\beta = \sum_i z_i^\beta \Delta R^k$$

- PC_3: fragmentation and energy dispersion

$$S_{\text{frag}} = -\sum_i z_i \log z_i$$

- $\text{PC}_{4,5}$: charge information etc

$$E_Q = \frac{E_{\text{charged}}}{E_{\text{jet}}} \qquad \text{and} \qquad A^\perp = S_{\text{frag}} \frac{C_{0.1}}{C_{0.05}} - 0.03 \cdot n_{\text{pf}} + 1.95 w_{\text{pf}}^\perp$$

$\rightarrow$ Latent distributions learn physics

ParticleNet beyond PCA

Disentangled latent classifier

- learning compressed, decorrelated representation

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} |x_i - \hat{x}_i|^2}_{\mathcal{L}_\text{reco}} + \underbrace{\sum_{i=1}^{N} \Big[y_i \log \sigma(z_i) + (1-y_i) \log(1-\sigma(z_i)) \Big]}_{\mathcal{L}_\text{class}} + \underbrace{\sum_{j \neq k} \Big[\text{Cov}(z_j, z_k) \Big]^2}_{\mathcal{L}_\text{disentangle}}$$

$\rightarrow$ 5 latent dimensions plenty

Latent Dim	1	2	3	4
AUC	0.893(2)	0.9001(4)	0.9024(4)	0.9034(2)
$\text{rej}_{30\%}$	72(3)	77(3)	´ 95(5)	95(3)
ΔC	1.8(3)	0.93(5)	1.0(16)	0.9(15)

ParticleNet beyond PCA

Disentangled latent classifier

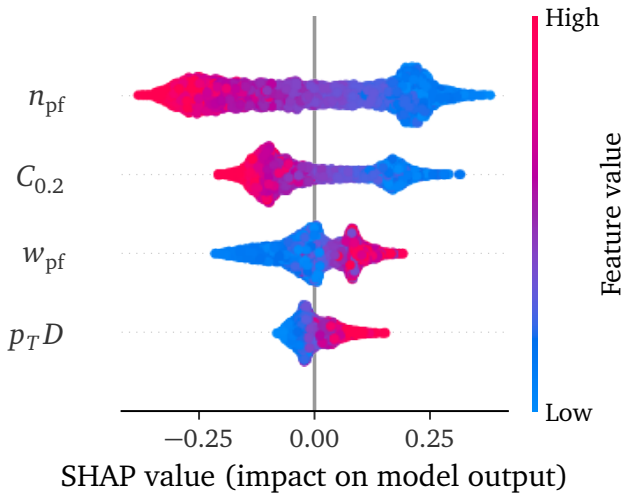
- learning compressed, decorrelated representation

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} |x_i - \hat{x}_i|^2}_{\mathcal{L}_\text{reco}} + \underbrace{\sum_{i=1}^{N} \Big[y_i \log \sigma(z_i) + (1-y_i) \log(1-\sigma(z_i)) \Big]}_{\mathcal{L}_\text{class}} + \underbrace{\sum_{j\neq k} \Big[\text{Cov}(z_j, z_k) \Big]^2}_{\mathcal{L}_\text{disentangle}}$$

$\rightarrow$ 5 latent dimensions plenty

Shapley variables

- pretty pictures, with weird patterns [quark



ParticleNet beyond PCA

Disentangled latent classifier

- learning compressed, decorrelated representation

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} |x_i - \hat{x}_i|^2}_{\mathcal{L}_\text{reco}} + \underbrace{\sum_{i=1}^{N} \Big[y_i \log \sigma(z_i) + (1-y_i) \log(1-\sigma(z_i)) \Big]}_{\mathcal{L}_\text{class}} + \underbrace{\sum_{j \neq k} \Big[\text{Cov}(z_j, z_k) \Big]^2}_{\mathcal{L}_\text{disentangle}}$$

$\rightarrow$ 5 latent dimensions plenty

Shapley variables

- pretty pictures, with weird patterns [quark jets: low multiplicity and low girth]

$\rightarrow$ Little insight from SHAP implementation

ParticleNet beyond PCA

Disentangled latent classifier

- learning compressed, decorrelated representation

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} |x_i - \hat{x}_i|^2}_{\mathcal{L}_\text{reco}} + \underbrace{\sum_{i=1}^{N} \Big[y_i \log \sigma(z_i) + (1-y_i) \log(1-\sigma(z_i)) \Big]}_{\mathcal{L}_\text{class}} + \underbrace{\sum_{j \neq k} \Big[\text{Cov}(z_j, z_k) \Big]^2}_{\mathcal{L}_\text{disentangle}}$$

$\rightarrow$ 5 latent dimensions plenty

Shapley variables

- pretty pictures, with weird patterns [quark jets: low multiplicity and low girth]

$\rightarrow$ Little insight from SHAP implementation

Symbolic Regression

- learn formula with given complexity
- classifier output not power series
- AUC and calibration double goal

$$p_\text{quark} = \tanh^3 \Big[0.55 \cdot C_{0.2} + 2 \Big(-0.02 \cdot r_\lambda \cdot \big(C_{0.2} \cdot p_T D \cdot S_\text{PID} \cdot S_\text{frag} - 0.25 \big) + 1 \Big)^3 \Big]$$

observables	model	AUC	$\text{Rej}_{30\%}$
$(n_\text{pf}, p_T D, C_{0.2}, r_\lambda, S_\text{PID}, S_\text{frag}, E_Q)$	MLP	0.872	66.87
	PySR	0.871	66.58

ParticleNet beyond PCA

Disentangled latent classifier

- learning compressed, decorrelated representation

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N} |x_i - \hat{x}_i|^2}_{\mathcal{L}_\text{reco}} + \underbrace{\sum_{i=1}^{N} \Big[y_i \log \sigma(z_i) + (1-y_i) \log(1-\sigma(z_i)) \Big]}_{\mathcal{L}_\text{class}} + \underbrace{\sum_{j \neq k} \Big[\text{Cov}(z_j, z_k) \Big]^2}_{\mathcal{L}_\text{disentangle}}$$

$\rightarrow$ 5 latent dimensions plenty

Shapley variables

- pretty pictures, with weird patterns [quark jets: low multiplicity and low girth]

$\rightarrow$ Little insight from SHAP implementation

Symbolic Regression

- learn formula with given complexity
- classifier output not power series
- AUC and calibration double goal

$$p_\text{quark} = \tanh^3 \Big[0.55 \cdot C_{0.2} + 2 \left(-0.02 \cdot r_\lambda \cdot \left(C_{0.2} \cdot p_T D \cdot S_\text{PID} \cdot S_\text{frag} - 0.25 \right) + 1 \right)^3 \Big]$$

$\rightarrow$ Formulas as physics regularizers?

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 $r \sim \mathcal{N}(r)$

$\rightarrow$ 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 $r \sim \mathcal{N}(r)$

$\rightarrow$ Learn Jacobian

Normalizing flow

- bijective mapping

$$\text{latent } r \sim p_{\text{latent}} \quad \underset{\leftarrow \overline{G}_\theta(x)}{\overset{G_\theta(r) \rightarrow}{\longleftrightarrow}} \quad \text{phase space } x \sim p_{\text{data}}$$

- tractable Jacobian

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

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

- likelihood loss

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

$\Rightarrow$ 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

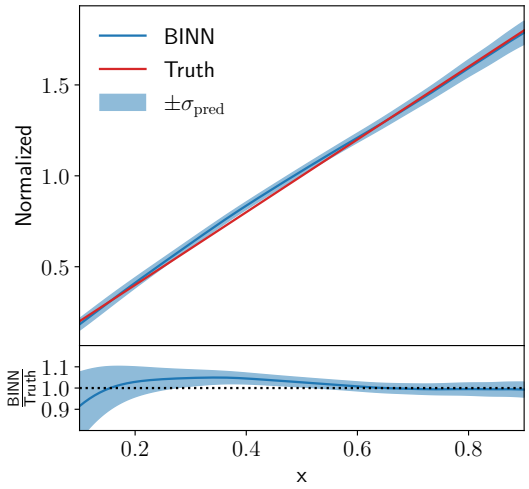
  allow weight distributions
  learn uncertainty map

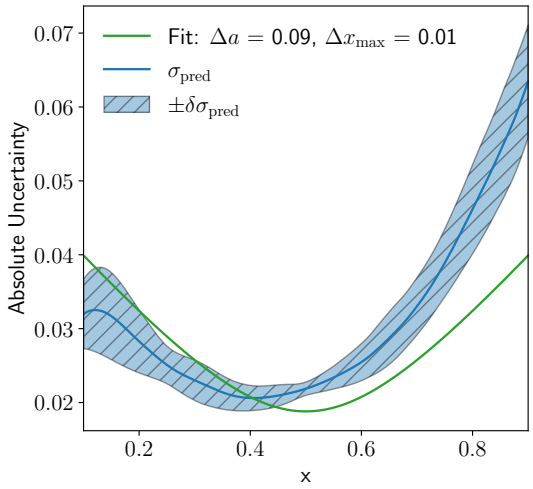
- 2D wedge ramp

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

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

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



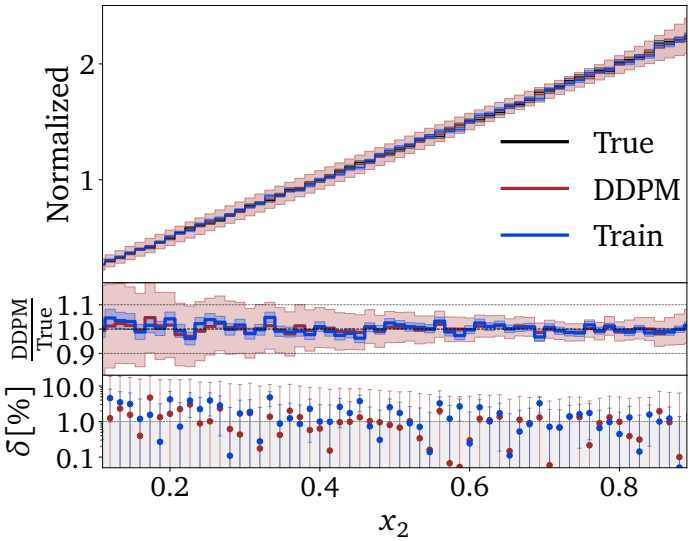


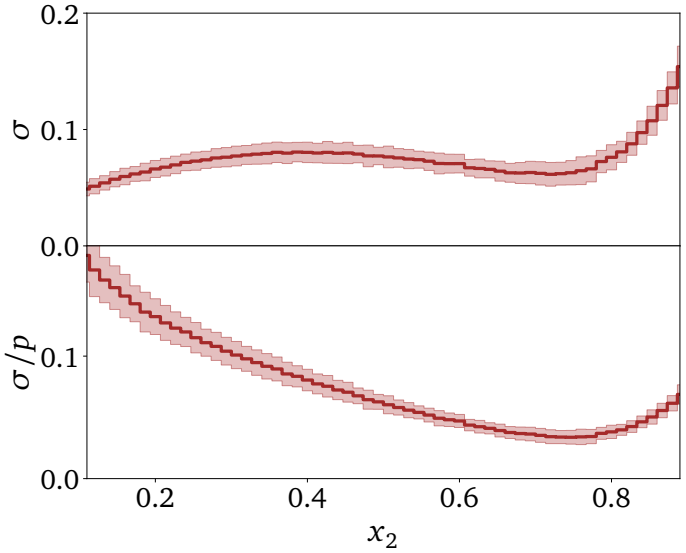
→ INNs just (non-parametric) fits

More architectures

Alternative architectures

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

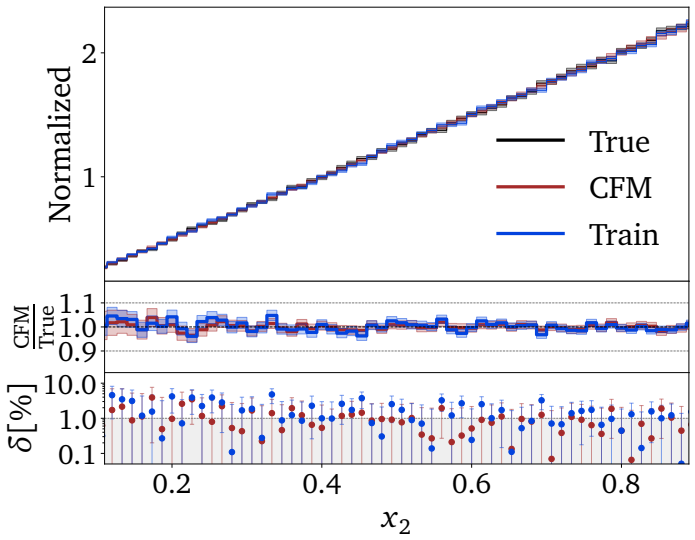


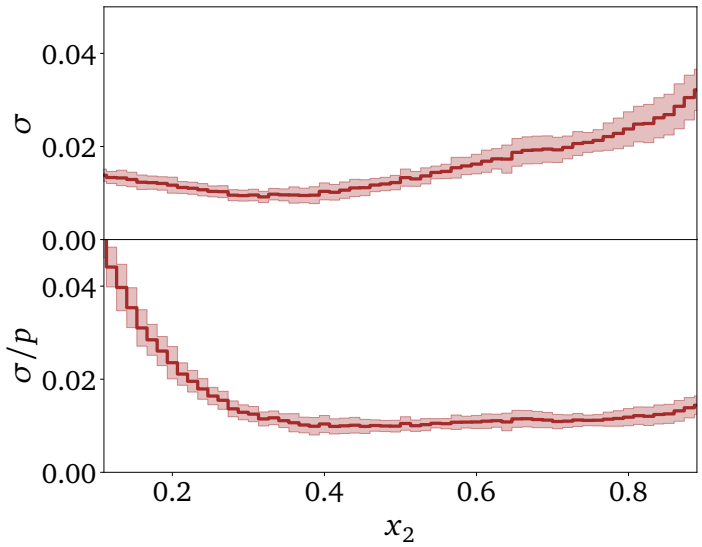


More architectures

Alternative architectures

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

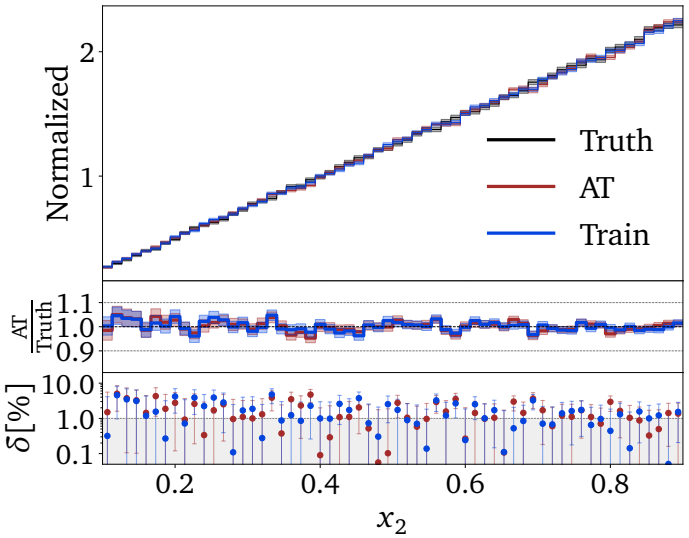


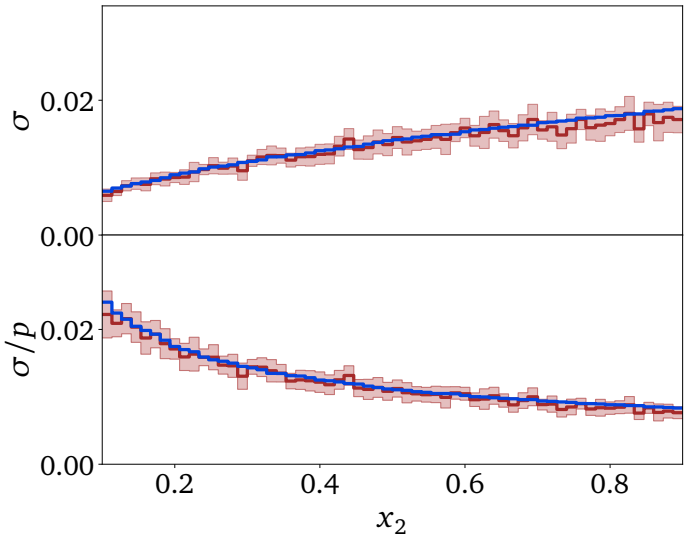


More architectures

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)

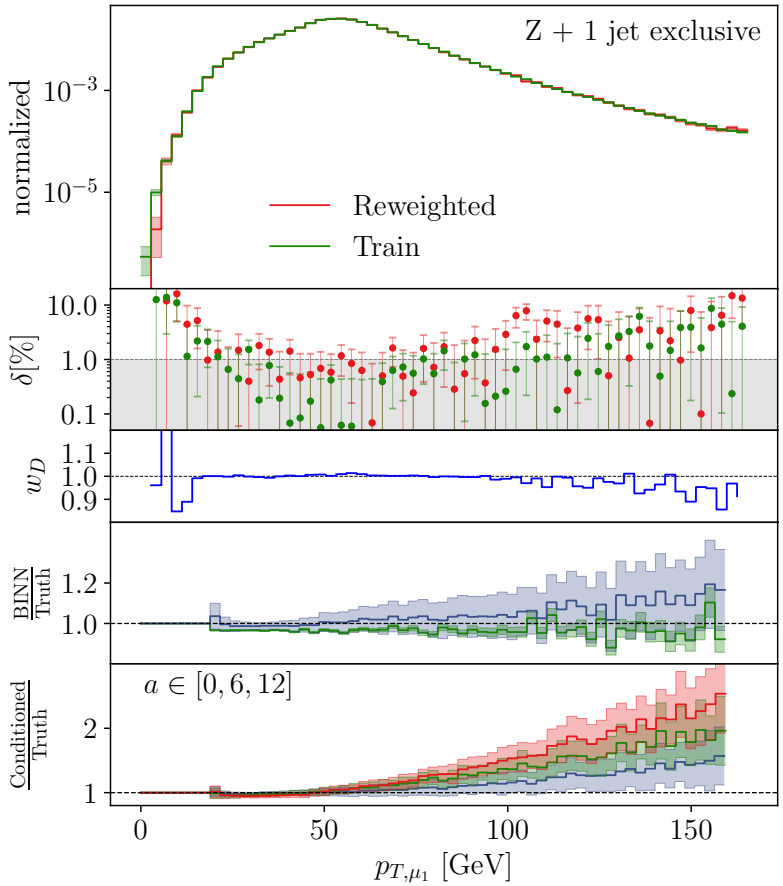
Z+jets events

- 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 $\{r, a\}$

$\rightarrow$ Precision event generation



Variational autoencoders

Low-dimensional latent representation [sufficient representation?]

- replace bijective mapping by latent encoding r

$$x \in \mathbb{R}^D \xrightarrow{p(r|x)} r \in \mathbb{R}^d \xrightarrow{p(x|r)} x' \in \mathbb{R}^D$$

- given decoder $p_\text{model}(x|r)$ learn encoder

$$p_\text{model}(r|x) \approx p(r|x) \equiv \frac{p_\text{model}(x|r)\; p_\text{latent}(r)}{p_\text{data}(x)}$$

- in terms of KL-divergence again

$$\begin{aligned}
D_\text{KL}[p_\text{model}(r|x), p(r|x)] &= \left\langle \log \frac{p_\text{model}(r|x)}{p(r|x)} \right\rangle_{p_\text{model}(r|x)} \\
&= \left\langle \log p_\text{model}(r|x) - \log \frac{p_\text{model}(x|r) p_\text{latent}(r)}{p_\text{data}(x)} \right\rangle_{p_\text{model}(r|x)} \\
&= -\left\langle \log p_\text{model}(x|r) \right\rangle_{p_\text{model}(r|x)} + \left\langle \log p_\text{model}(r|x) - \log p_\text{latent}(r) \right\rangle_{p_\text{model}(r|x)} + \dots \\
&= -\left\langle \log p_\text{model}(x|r) \right\rangle_{p_\text{model}(r|x)} + D_\text{KL}[p_\text{model}(r|x), p_\text{latent}(r)] + \dots
\end{aligned}$$

- autoencoder loss evaluated over data

$$\mathcal{L}_\text{VAE} = \left\langle - \left\langle \log p_\text{model}(x|r) \right\rangle_{p_\text{model}(r|x)} + \beta_\text{KL} D_\text{KL}[p_\text{model}(r|x), p_\text{latent}(r)] \right\rangle_{p_\text{data}}$$

$\to$ Reconstruction loss plus regularization

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

Testing generative networks

Compare network to training/test data

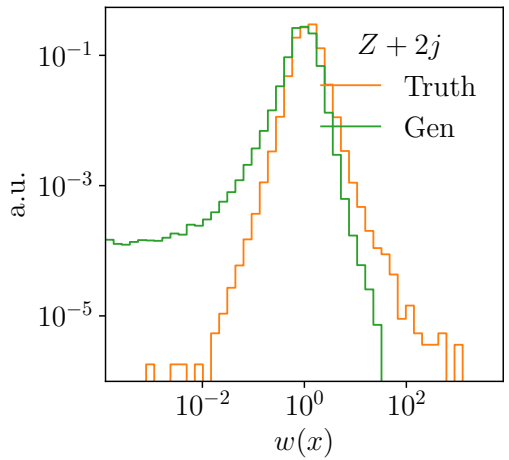
- supervised: histogram deviation from truth
- unsupervised density $\rightarrow$ histogram discriminator

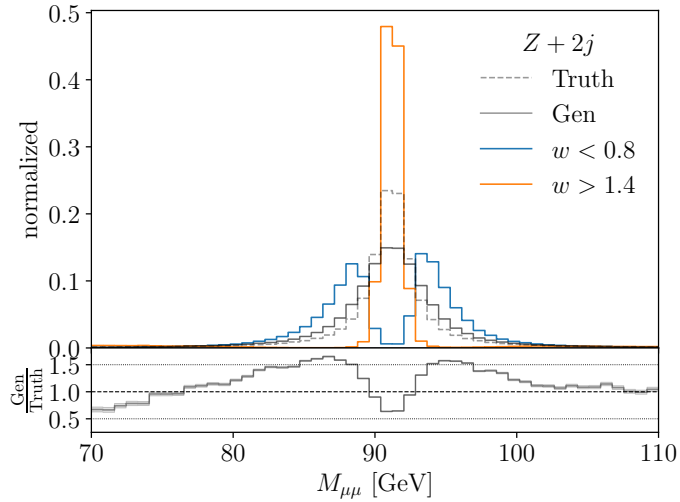
$$w(x_i) = \frac{D(x_i)}{1 - D(x_i)} = \frac{p_{\text{data}}(x_i)}{p_{\text{model}}(x_i)}$$

$\rightarrow$ Using interpretable phase space

Applied to LHC events over phase space

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





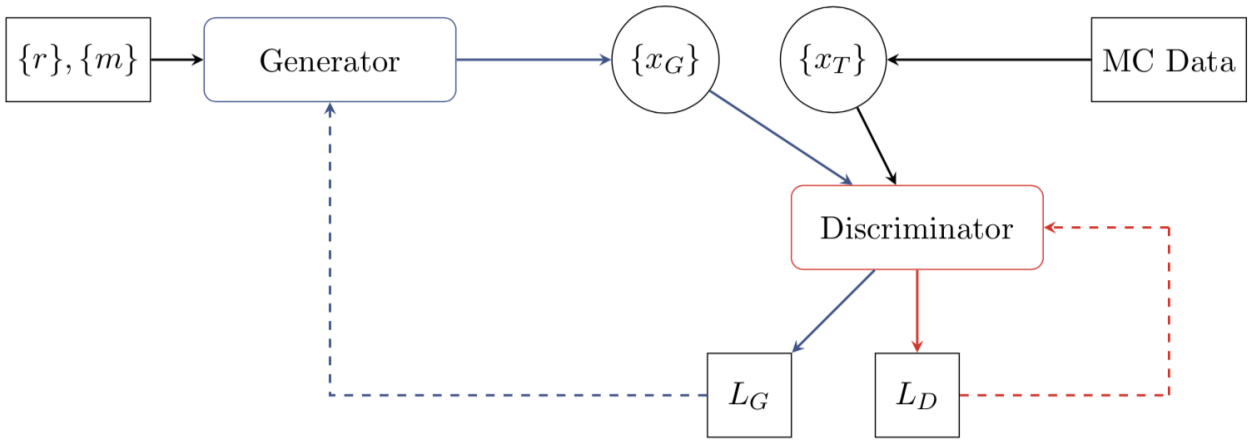
$\rightarrow$ Local failure modes

Generative adversarial networks

Generating events

- training: true events $p_{\text{data}}(x)$
  output: generated events $r \to x \sim p_{\text{model}}(x)$
- discriminator/classifier
$$D(x) \in [0,1] \to \begin{cases} 0 & \text{true data} \\ 1 & \text{generated} \end{cases}$$
- classifier loss defining Nash equilibrium
$$\mathcal{L}_D \equiv \mathcal{L}_{\text{class}} = -\big\langle \log D(x) \big\rangle_{p_{\text{data}}} - \big\langle \log(1 - D(x)) \big\rangle_{p_{\text{model}}} \to -2 \log \frac{1}{2}$$
- generator loss [only D needed]
$$\mathcal{L}_G = \big\langle -\log D(x) \big\rangle_{p_{\text{model}}} \to -\log \frac{1}{2}$$

$\Rightarrow$ statistically independent copy of training data



Equivariant event generation

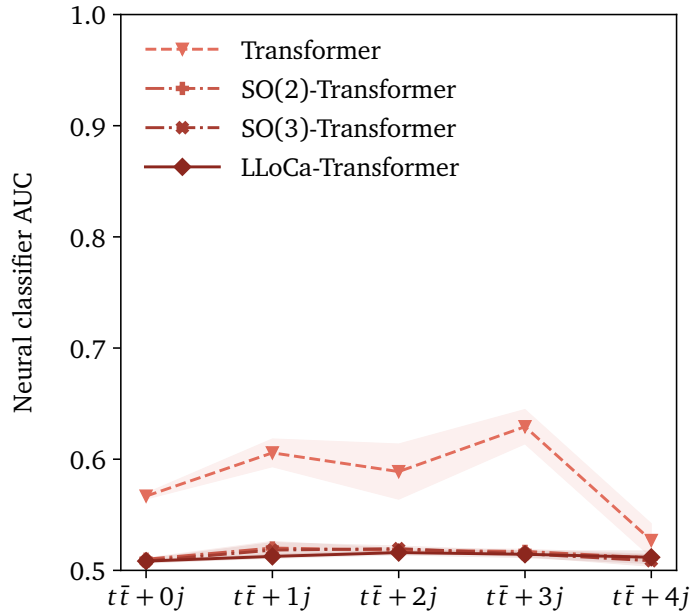
State of the art

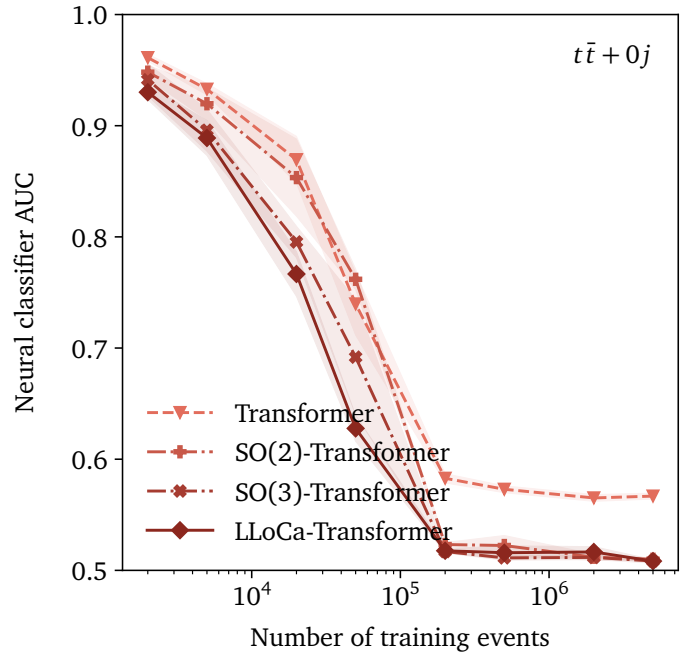
- conditional flow matching or continuous normalizing flow
- encode velocity using neural network

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

- 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

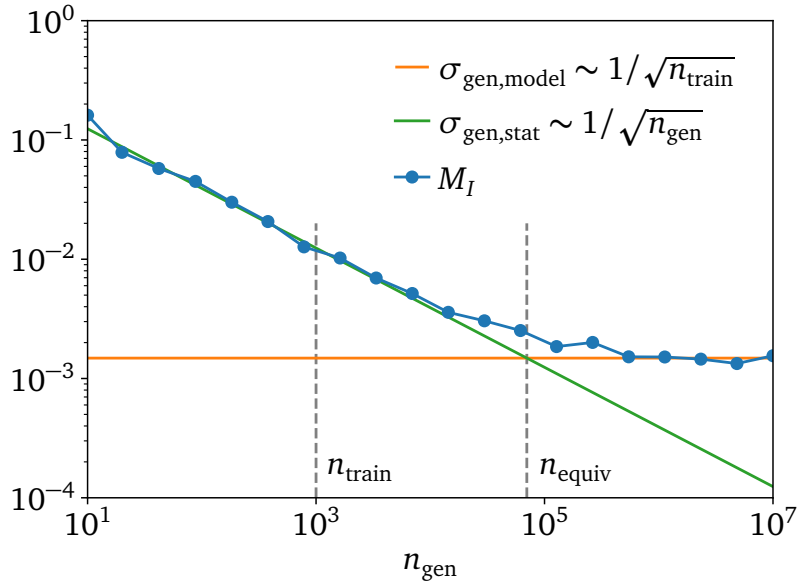
$$D_{\text{true}}^{n_{\text{train}}} \sim p_{\text{true}} \approx p_{\text{gen}}(x) \quad \Rightarrow \quad p_{\text{gen}}(x) \overset{?}{\sim} p_{\text{true}}(x)$$

- maximum meaningful equivalent number of events

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

- compare two densities with two uncertainties

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



Generative amplification

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

- learned phase space density

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

- maximum meaningful equivalent number of events

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

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

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

$$\Rightarrow \quad M_I\left[D_{\text{gen}}^{n_{\text{gen}}}, p_{\text{true}}\right] = \left\langle \left[\bar{I}(D_{\text{gen}}^{n_{\text{gen}}}) - I(p_{\text{true}})\right]^2 \right\rangle$$

$$= \begin{cases} \sigma^2_{\text{stat}}(n_{\text{gen}}) & p_{\text{gen}} = p_{\text{true}} \\ \sigma^2_{\text{stat}}(n_{\text{gen}}) + \sigma^2_{\text{model}}(p_{\text{gen}}, p_{\text{true}}) & p_{\text{gen}} \neq p_{\text{true}} \end{cases} .$$

- binomial distribution to extrapolate

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

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

$\to$ Needing model uncertainty, eventually BNN

Extrapolating ISR

Universal QCD jet radiation [$Z + 1 \ldots 8$ jets]

- from n to $n+1$ jets

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

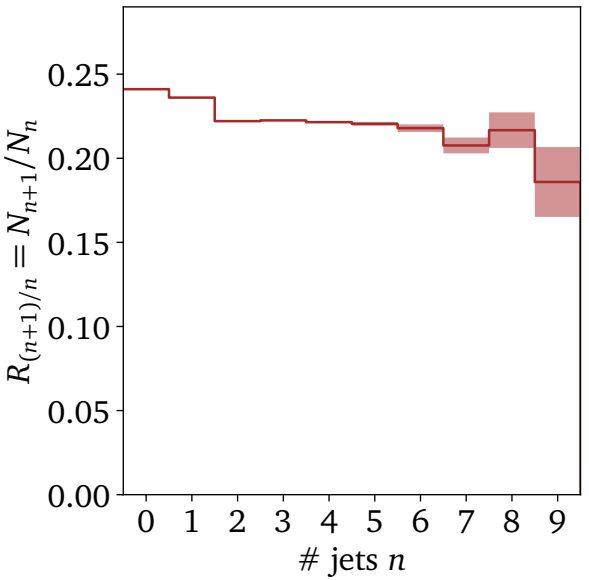
- large scale drop: Poisson scaling

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

- democratic scales: staircase scaling

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

$\rightarrow$ Universal pattern learnable?



Extrapolating ISR

Universal QCD jet radiation [$Z + 1 \ldots 8$ jets]

- democratic scales: staircase scaling

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

$\rightarrow$ Universal pattern learnable?

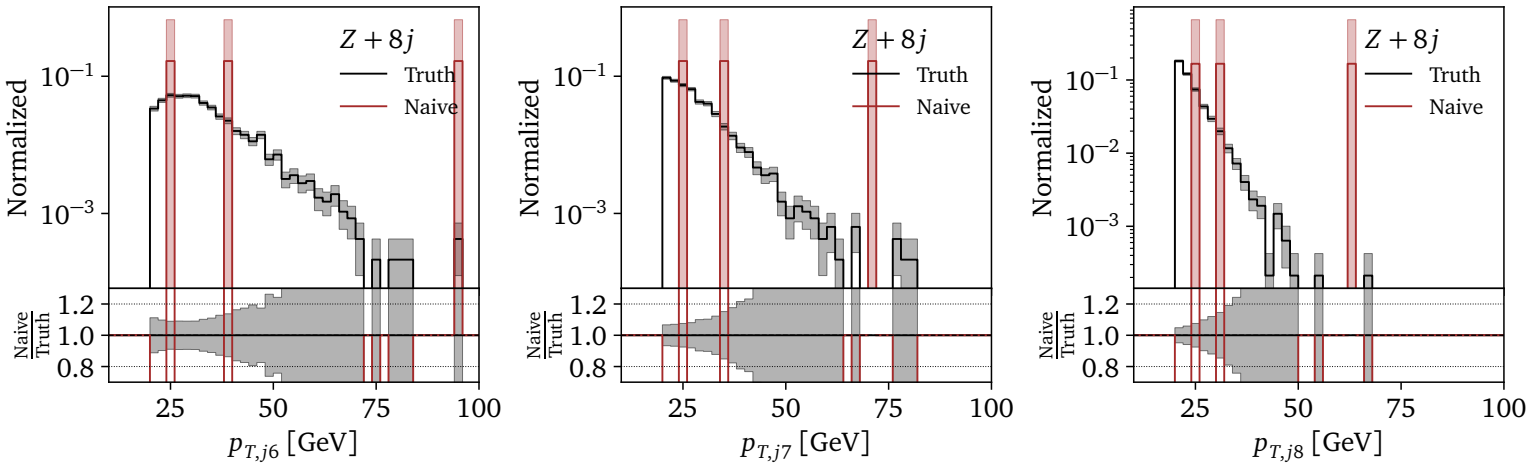
Autoregressive transformer

- factorized probability and loss function

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

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

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



Extrapolating ISR

Universal QCD jet radiation [$Z + 1 \ldots 8$ jets]

- democratic scales: staircase scaling

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

$\rightarrow$ Universal pattern learnable?

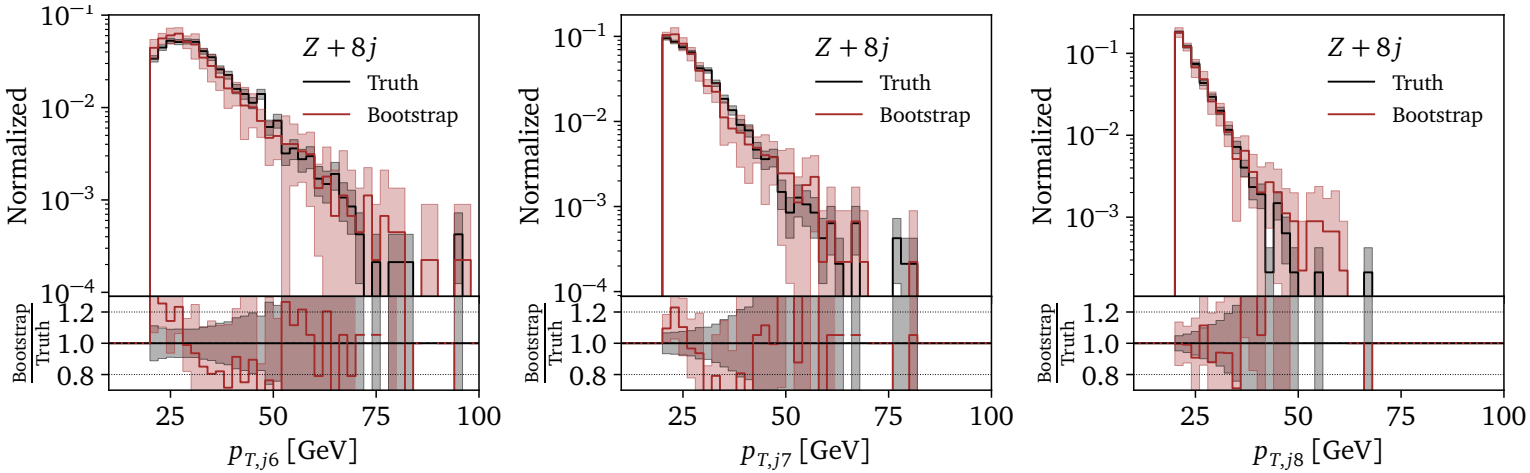
Autoregressive transformer

- factorized probability and loss function

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

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

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



Extrapolating ISR

Universal QCD jet radiation [$Z + 1 \ldots 8$ jets]

- democratic scales: staircase scaling

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

$\rightarrow$ Universal pattern learnable?

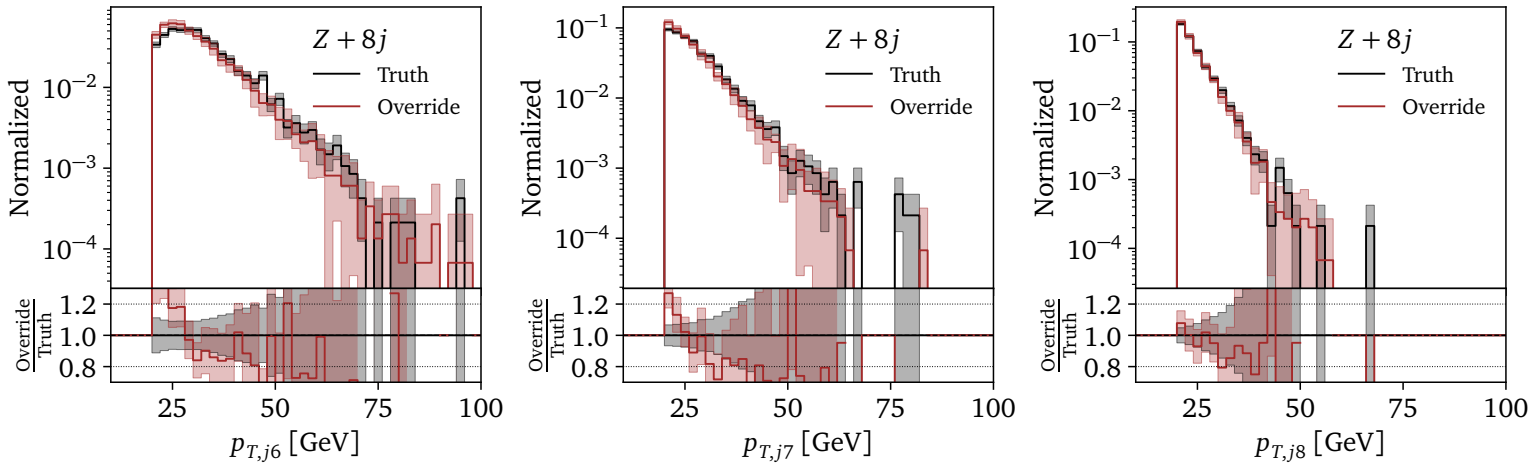
Autoregressive transformer

- factorized probability and loss function

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

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

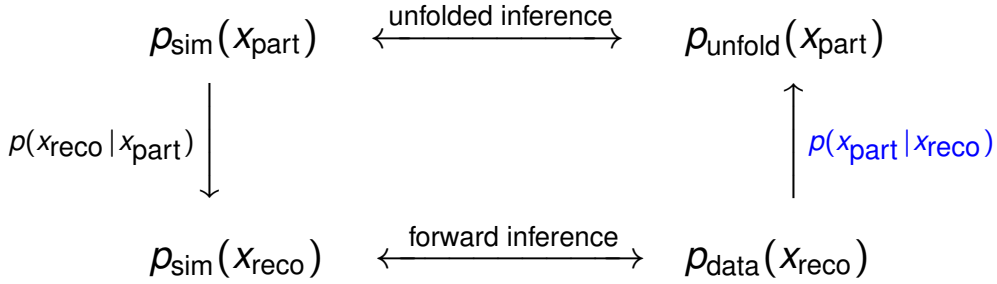
$\rightarrow$ full extrapolation with right latent representation



Deconvolution, unfolding, generative inference

Basic structure

- four phase space distributions



- two conditional probabilities

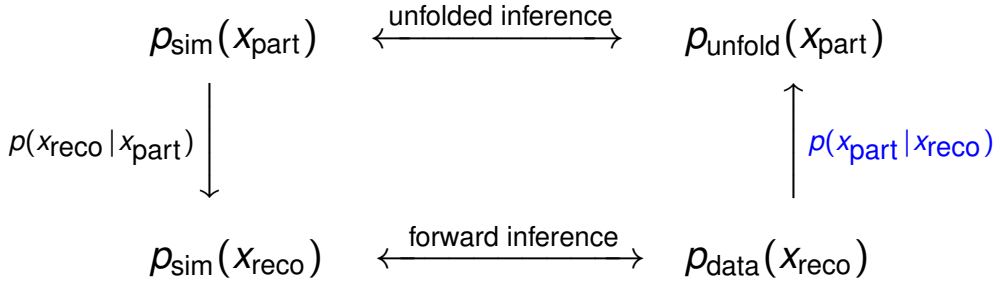
$$p(x_{\text{part}}|x_{\text{reco}}) = p(x_{\text{reco}}|x_{\text{part}}) \times \frac{p_{\text{sim}}(x_{\text{part}})}{p_{\text{sim}}(x_{\text{reco}})}$$

$\rightarrow$ Unbinned and high-dimensional unfolding

Deconvolution, unfolding, generative inference

Basic structure

- four phase space distributions



- conditional probabilities from joint simulations

$$p(x_1\,|\,x_2) \sim \frac{p(x_1, x_2)}{p(x_2)} \qquad \text{in either direction}$$

$\rightarrow$ Unbinned and high-dimensional unfolding

Deconvolution, unfolding, generative inference

Basic structure

- four phase space distributions

$$\begin{array}{ccc} p_{\text{sim}}(x_{\text{part}}) & \xleftrightarrow{\text{unfolded inference}} & p_{\text{unfold}}(x_{\text{part}}) \\ \Big\downarrow p(x_{\text{reco}}\,|\,x_{\text{part}}) & & \Big\uparrow p(x_{\text{part}}\,|\,x_{\text{reco}}) \\ p_{\text{sim}}(x_{\text{reco}}) & \xleftrightarrow{\text{forward inference}} & p_{\text{data}}(x_{\text{reco}}) \end{array}$$

$\rightarrow$ Unbinned and high-dimensional unfolding

Training bias

- true mass m_t vs mass in simulation m_s

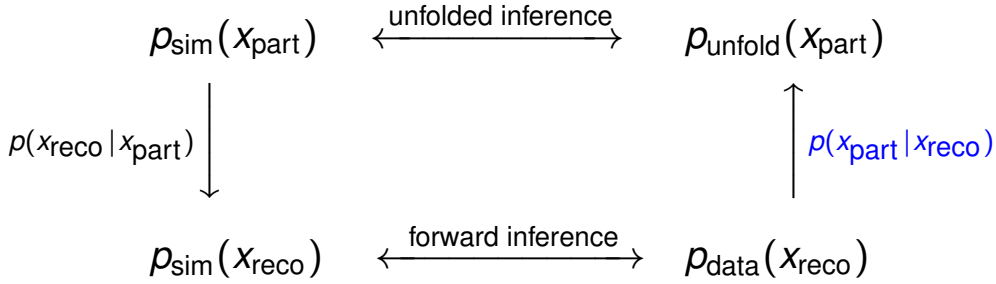
$$\begin{array}{ccc} p_{\text{sim}}(x_{\text{part}}\,|\,m_s) & \xleftrightarrow{\text{unfolded inference}} & p_{\text{unfold}}(x_{\text{part}}\,|\,m_s, m_t) \\ \Big\downarrow p(x_{\text{reco}}\,|\,x_{\text{part}}, m_s) & & \Big\uparrow p(x_{\text{part}}\,|\,x_{\text{reco}}, m_s) \\ p_{\text{sim}}(x_{\text{reco}}\,|\,m_s) & \xleftrightarrow{\text{forward inference}} & p_{\text{data}}(x_{\text{reco}}\,|\,m_t) \end{array}$$

$\rightarrow$ Solved by 'Bayesian' iterative unfolding

Deconvolution, unfolding, generative inference

Basic structure

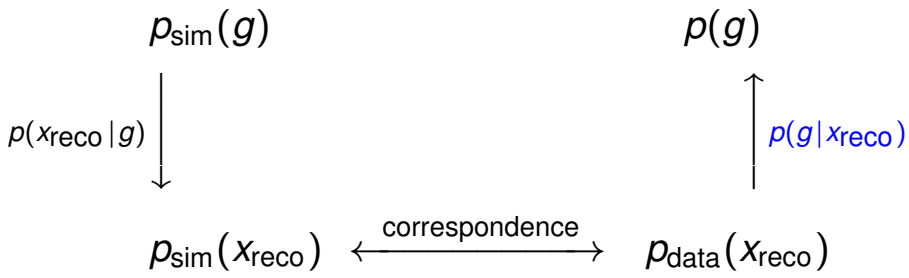
- four phase space distributions



$\rightarrow$ Unbinned and high-dimensional unfolding

Generative inference [cosmology, gravitational waves]

- invert into model space



- conditional generation of correlated posterior

$\rightarrow$ Amortized inference

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

Modern Machine Learning for LHC Physicists

Tilman Plehn[a*], Anja Butter[a,b], Barry Dillon[a],
Theo Heimel[a], 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]