Inference-Aware Neural Optimisation (INFERNO)

ML-INFN Meeting

Pablo de Castro, Tommaso Dorigo,
Lukas Layer, Giles Strong

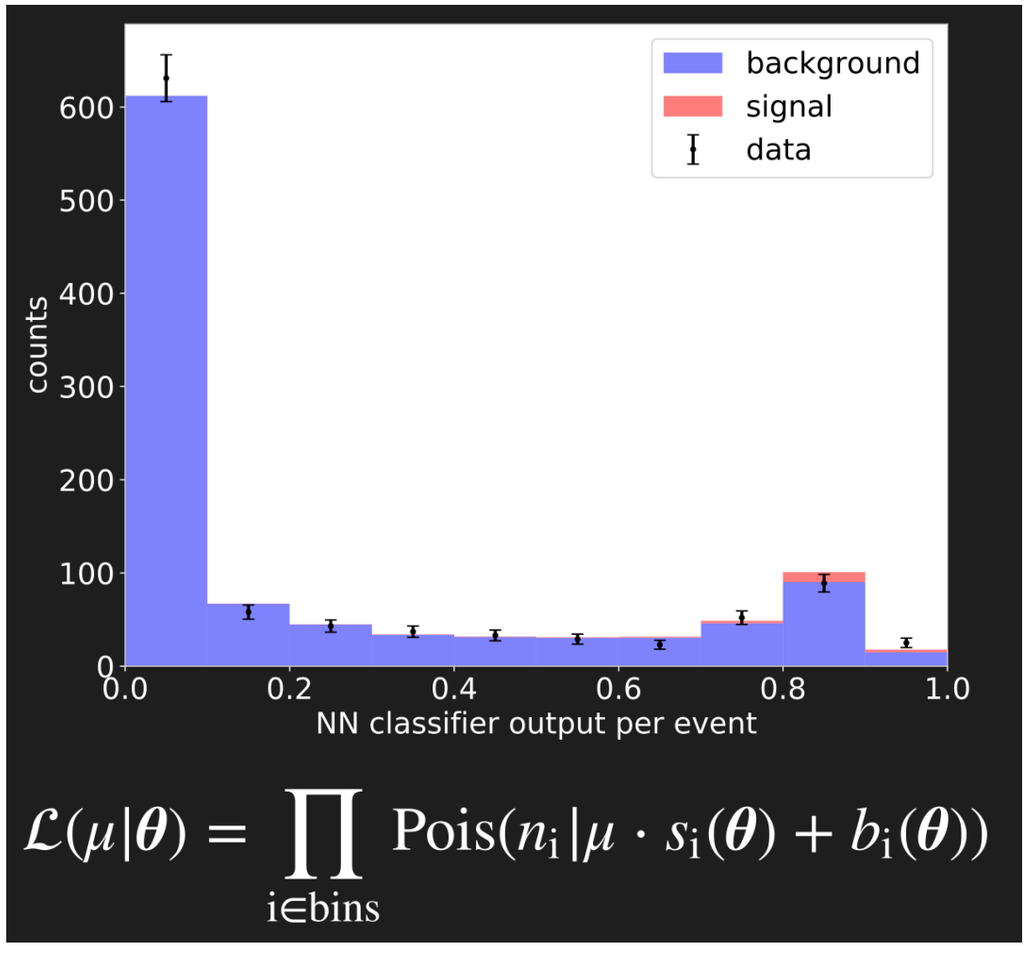
Problem Statement

Our data is high dimensional.
How do we choose the best summary statistics for the inference of the parameter of interest?

Classical approach in HEP

- **Signal vs. background classification** with simulated samples by minimizing **Binary Cross-Entropy** of **Neural Networks** to approximate the optimal Likelihood Ratio

- Inference with **histograms** by building binned non-parametric **Poisson counts likelihood**



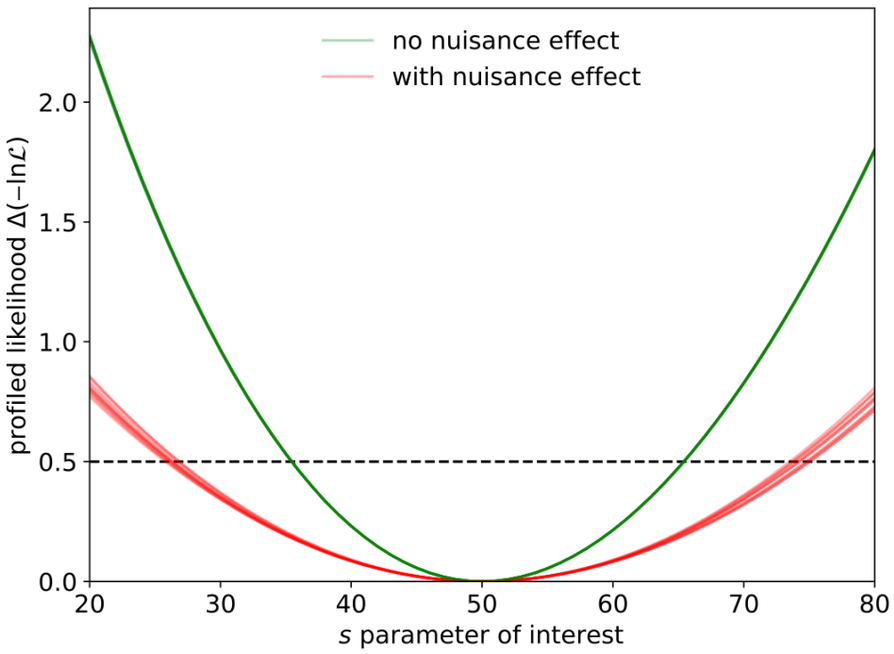
Problem Statement

Our data is high dimensional.
How do we choose the best summary statistics for the inference of the parameter of interest?

- Problem: **simulations** are **imperfect** → lack of knowledge for inference accounted by additional **nuisance parameters**

- Neglecting nuisance parameters in the training process leads to **larger** measurement **uncertainties**

- **Upper limit** of ML usefulness in LHC analyses



Idea: include nuisance parameters in the loss function of a Neural Network and directly minimize variance of POI

INFERNO

Overview

- **Inference-Aware Neural Optimization** developed by **Pablo de Castro** and **Tommaso Dorigo:** P. de Castro Manzano, T. Dorigo, “INFERNO: Inference-Aware Neural Optimization”, Comp. Phys. Commun. 244 (2019) 170; https://doi.org/10.1016/j.cpc.2019.06.007

- **Objective**: optimize statistical **inference** in the presence of **nuisance parameters** by learning non-linear **summary statistics** with neural networks via minimization of an **inference-motivated loss**

- Code written in **TF 1.x** (https://github.com/pablodecm/paper-inferno) → key elements reproduced in **TF 2.x** (https://github.com/llayer/inferno)

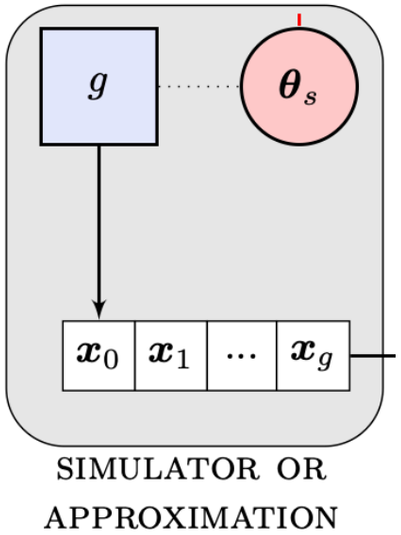
Summary statistics with INFERNO

- “Standard HEP summary statistics” **histograms** are not differentiable → **softmax** as differentiable approximation

- **Nuisance parameters approximated** by changes of mixture coefficients, translations of a subset of features, or conditional density ratio re-weighting → restricted to **likelihood-free inference** problems

$$\hat{s}_i(D;\phi) = \sum_{x \in D} \frac{e^{f_i(x;\phi)/\tau}}{\sum_{j=0}^{b} e^{f_j(x;\phi)/\tau}}$$

Algorithm

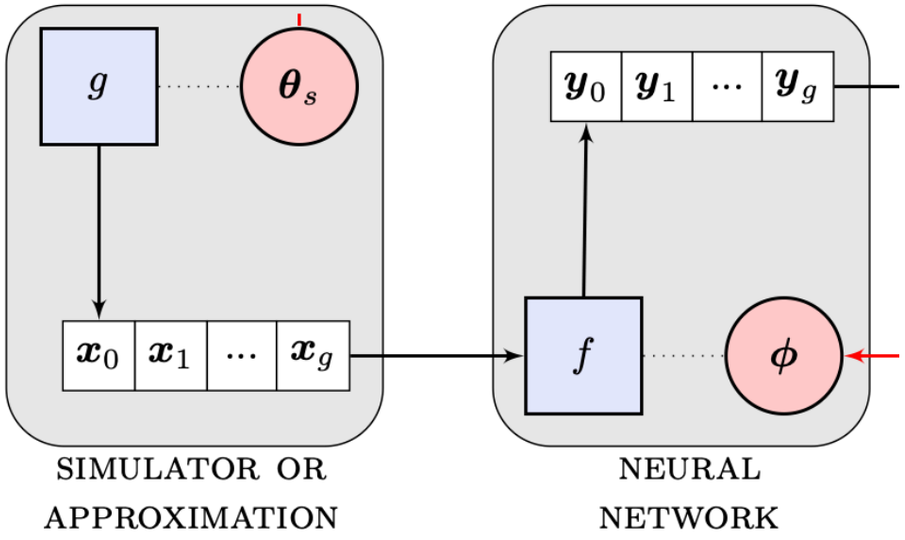


Algorithm 1 Inference-Aware Neural Optimisation.

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D; \boldsymbol{\phi})$.

Taken from https://arxiv.org/pdf/1806.04743.pdf

Algorithm



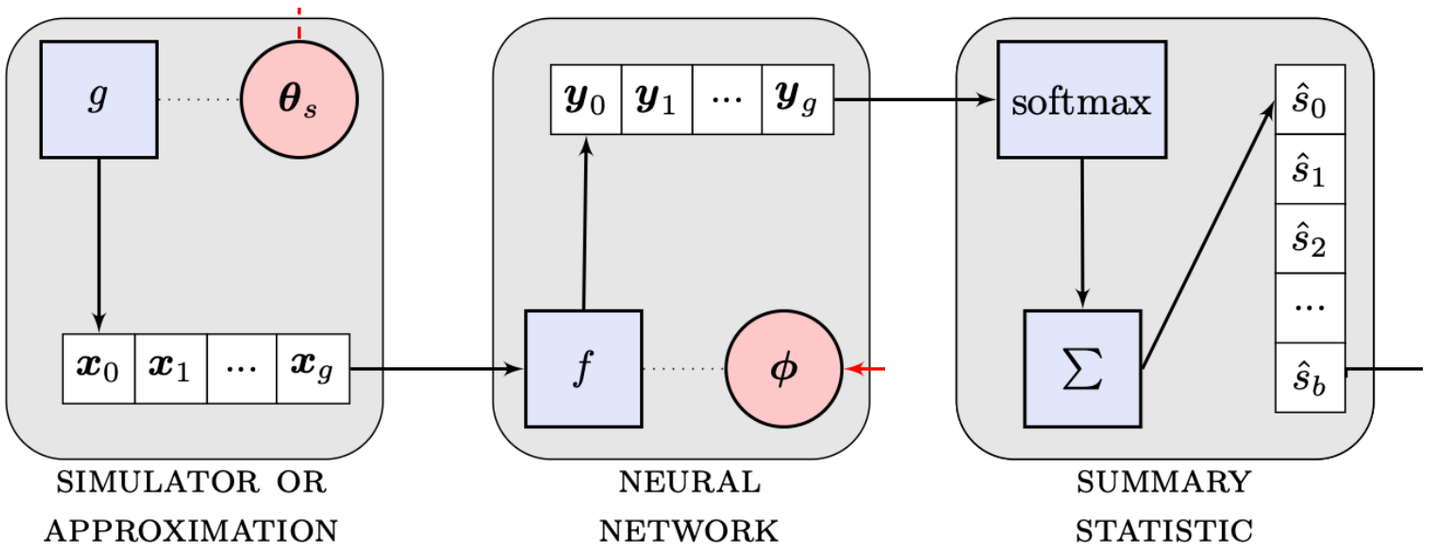
Algorithm 1 Inference-Aware Neural Optimisation.

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D; \boldsymbol{\phi})$.

1: **for** $i = 1$ to N **do**
2: Sample a representative mini-batch G_s from $g(\boldsymbol{\theta}_s)$.

Taken from https://arxiv.org/pdf/1806.04743.pdf

Algorithm



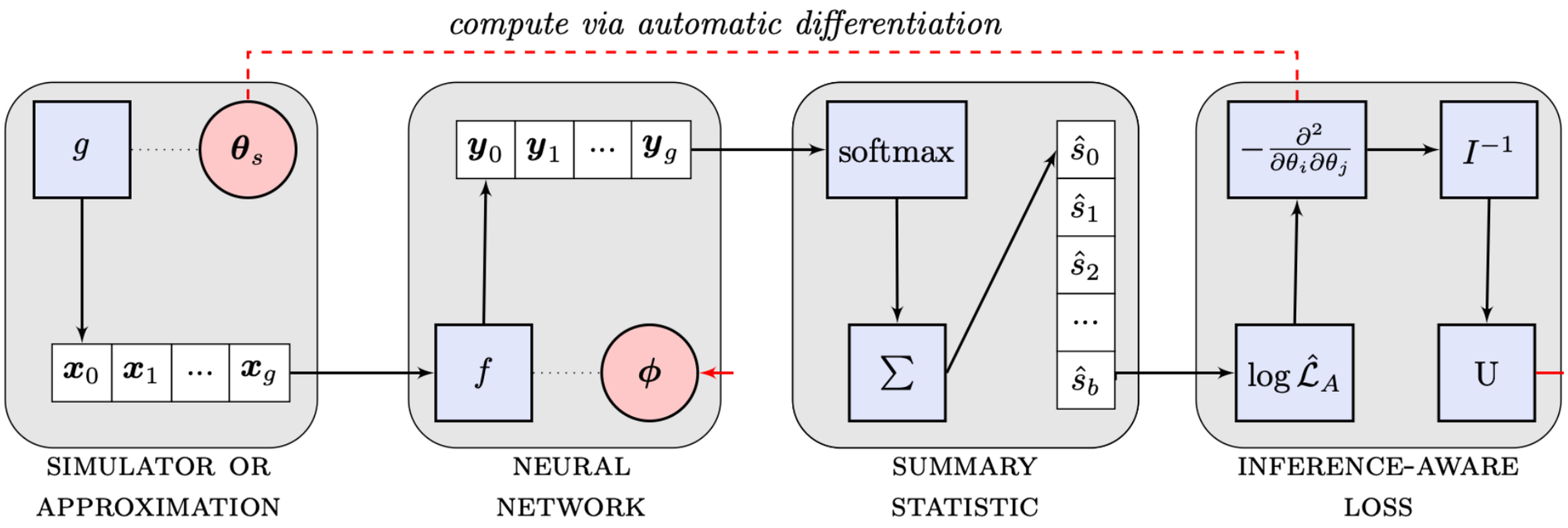
Algorithm 1 Inference-Aware Neural Optimisation.

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D; \boldsymbol{\phi})$.

1: **for** $i = 1$ to N **do**
2: Sample a representative mini-batch G_s from $g(\boldsymbol{\theta}_s)$.
3: Compute differentiable summary statistic $\hat{\boldsymbol{s}}(G_s; \boldsymbol{\phi})$.

Taken from https://arxiv.org/pdf/1806.04743.pdf

Algorithm



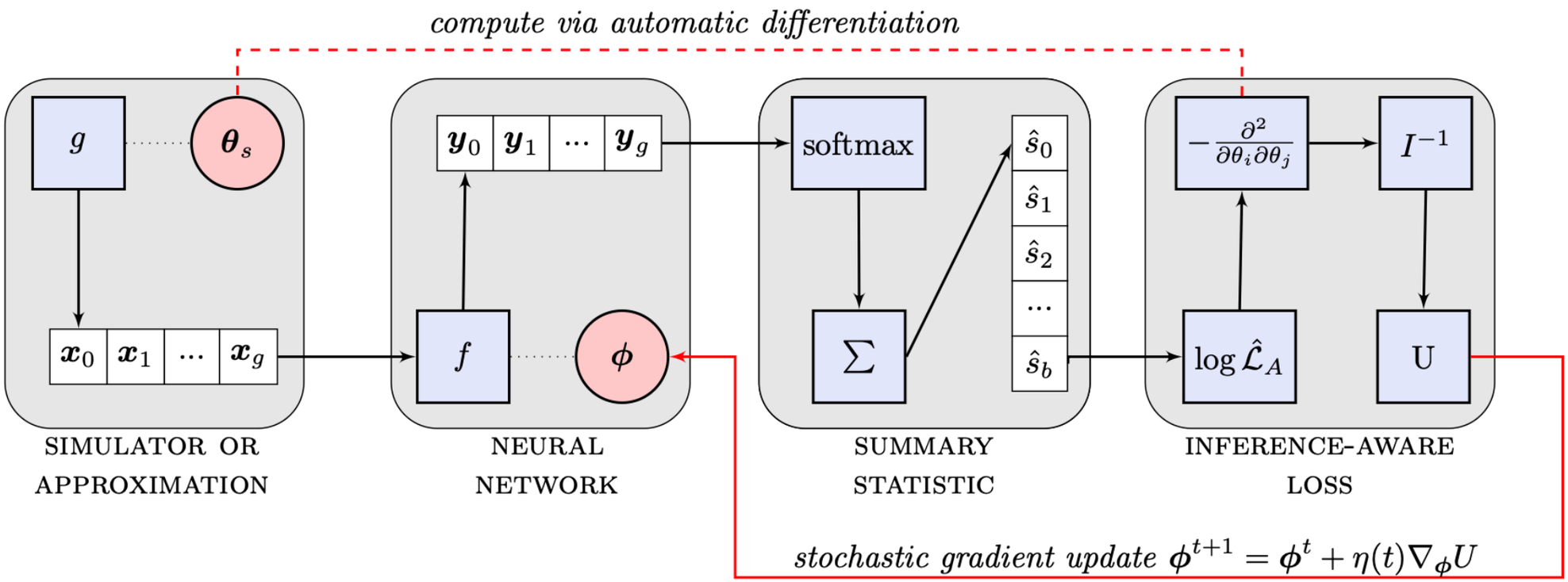
Algorithm 1 Inference-Aware Neural Optimisation.

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D; \boldsymbol{\phi})$.

for $i = 1$ to N **do**
  Sample a representative mini-batch G_s from $g(\boldsymbol{\theta}_s)$.
  Compute differentiable summary statistic $\hat{\mathbf{s}}(G_s; \boldsymbol{\phi})$.
  Construct Asimov likelihood $\mathcal{L}_A(\boldsymbol{\theta}, \boldsymbol{\phi})$.
  Get information matrix inverse $I(\boldsymbol{\theta})^{-1} = \boldsymbol{H}_{\boldsymbol{\theta}}^{-1}(\log \mathcal{L}_A(\boldsymbol{\theta}, \boldsymbol{\phi}))$.
  Obtain loss $U = I_{kk}^{-1}(\boldsymbol{\theta}_s)$.

Taken from https://arxiv.org/pdf/1806.04743.pdf

Algorithm



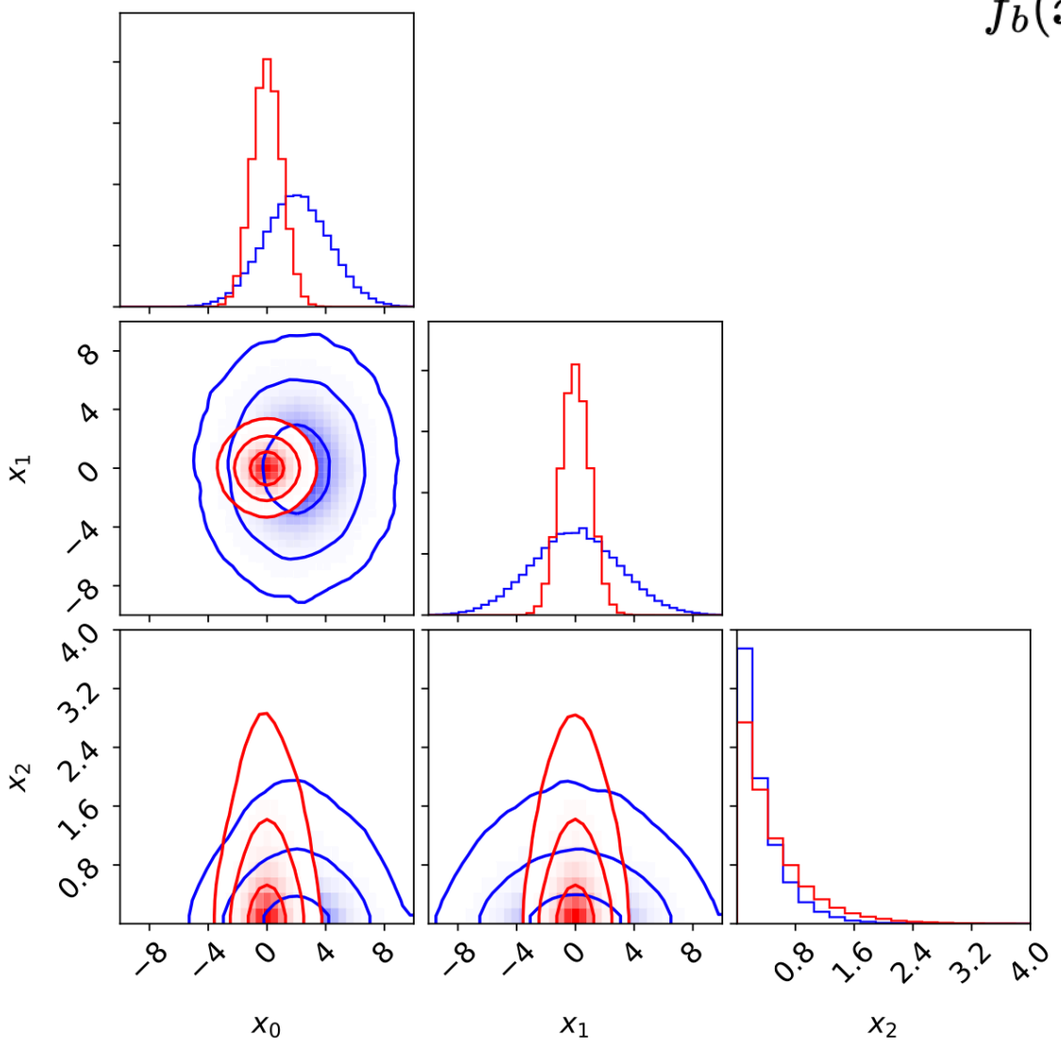
Algorithm 1 Inference-Aware Neural Optimisation.

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D;\boldsymbol{\phi})$.

1: **for** $i = 1$ to N **do**
2: Sample a representative mini-batch G_s from $g(\boldsymbol{\theta}_s)$.
3: Compute differentiable summary statistic $\hat{\mathbf{s}}(G_s;\boldsymbol{\phi})$.
4: Construct Asimov likelihood $\mathcal{L}_A(\boldsymbol{\theta},\boldsymbol{\phi})$.
5: Get information matrix inverse $I(\boldsymbol{\theta})^{-1} = \boldsymbol{H}_{\boldsymbol{\theta}}^{-1}(\log\mathcal{L}_A(\boldsymbol{\theta},\boldsymbol{\phi}))$.
6: Obtain loss $U = I_{kk}^{-1}(\boldsymbol{\theta}_s)$.
7: Update network parameters $\boldsymbol{\phi} \rightarrow \text{SGD}(\nabla_{\boldsymbol{\phi}}U)$.
8: **end for**

Taken from https://arxiv.org/pdf/1806.04743.pdf

Application to 3D Synthetic Mixture model



Taken from https://arxiv.org/pdf/1806.04743.pdf

Model

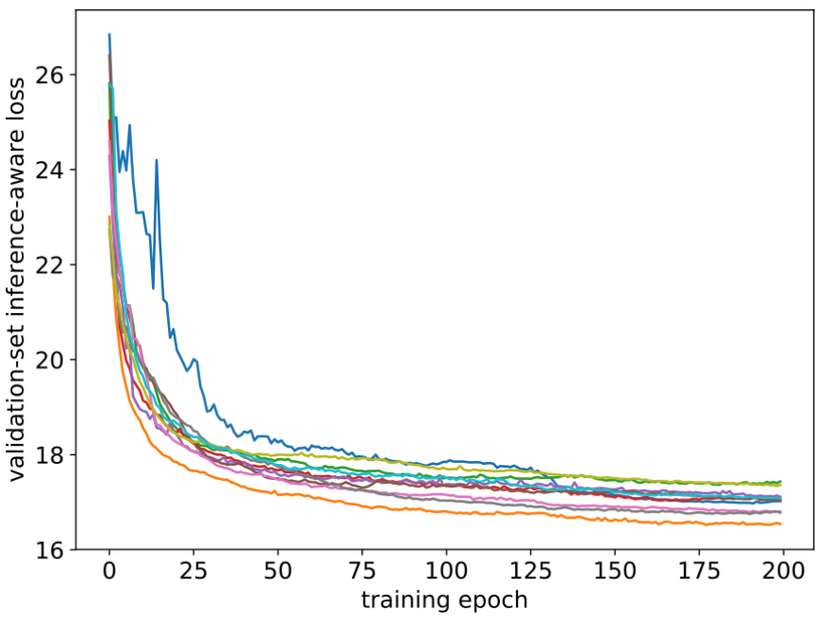
$$f_b(\boldsymbol{x}|r,\lambda) = \mathcal{N}\left((x_0,x_1)\,\middle|\,(2+r,0),\begin{bmatrix}5 & 0\\ 0 & 9\end{bmatrix}\right) Exp(x_2|\lambda)$$

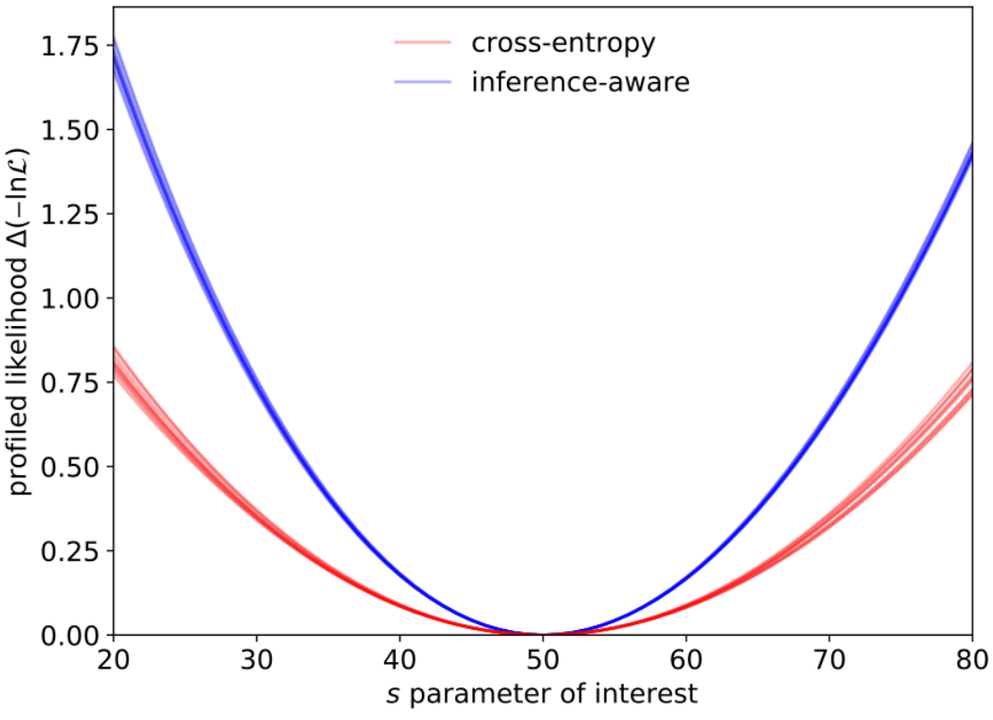
$$f_s(\boldsymbol{x}) = \mathcal{N}\left((x_0,x_1)\,\middle|\,(1,1),\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right) Exp(x_2|2)$$

$$p(\boldsymbol{x}|\mu,r,\lambda) = (1-\mu)f_b(\boldsymbol{x}|r,\lambda) + \mu f_s(\boldsymbol{x})$$

- 3D Mixture model: 2D Multivariate Normal and 1D Exponential
- Signal distribution is fully specified
- Background distribution depends on the parameters r (shift of the 2D Normal mean) and λ (rate of the exponential)

Results for the synthetic model





Taken from https://arxiv.org/pdf/1806.04743.pdf

Comparison of inference with INFERNO summary statistics and classical binary cross-entropy summary statistics

- **Confidence intervals** considerably **narrower** with INFERNO than those using BCE and closer to those expected when using the true model
- **Improvement** over standard classification increases with **more nuisance parameters**
- The inference-aware summary statistics learnt for $\boldsymbol{\theta}_S$ work well when $\boldsymbol{\theta}_{true} \neq \boldsymbol{\theta}_S$ in the range of variation explored.

Benchmarks

- Several inference problems are considered based on the 3D benchmark mixture model
- Systematic comparison, shows that **INFERNO** clearly **outperforms any classifier** (even optimal Bayes) when nuisance parameters are relevant

Table 1: Expected uncertainty on the parameter of interest s for each of the inference benchmarks considered using a cross-entropy trained neural network model, INFERNO customised for each problem and the optimal classifier and likelihood based resuls.

	Benchmark 0	Benchmark 1	Benchmark 2	Benchmark 3	Benchmark 4
NN classifier	$14.99^{+0.02}_{-0.00}$	$18.94^{+0.11}_{-0.05}$	$23.94^{+0.52}_{-0.17}$	$21.54^{+0.27}_{-0.05}$	$26.71^{+0.56}_{-0.11}$
INFERNO 0	$\mathbf{15.51^{+0.09}_{-0.02}}$	$18.34^{+5.17}_{-0.51}$	$23.24^{+6.54}_{-1.22}$	$21.38^{+3.15}_{-0.69}$	$26.38^{+7.63}_{-1.36}$
INFERNO 1	$15.80^{+0.14}_{-0.04}$	$\mathbf{16.79^{+0.17}_{-0.05}}$	$21.41^{+2.00}_{-0.53}$	$20.29^{+1.20}_{-0.39}$	$24.26^{+2.35}_{-0.71}$
INFERNO 2	$15.71^{+0.15}_{-0.04}$	$16.87^{+0.19}_{-0.06}$	$\mathbf{16.95^{+0.18}_{-0.04}}$	$16.88^{+0.17}_{-0.03}$	$18.67^{+0.25}_{-0.05}$
INFERNO 3	$15.70^{+0.21}_{-0.04}$	$16.91^{+0.20}_{-0.05}$	$16.97^{+0.21}_{-0.04}$	$\mathbf{16.89^{+0.18}_{-0.03}}$	$18.69^{+0.27}_{-0.04}$
INFERNO 4	$15.71^{+0.32}_{-0.06}$	$16.89^{+0.30}_{-0.07}$	$16.95^{+0.38}_{-0.05}$	$16.88^{+0.40}_{-0.05}$	$\mathbf{18.68^{+0.58}_{-0.07}}$
Optimal classifier	14.97	19.12	24.93	22.13	27.98
Analytical likelihood	14.71	15.52	15.65	15.62	16.89

Taken from https://arxiv.org/pdf/1806.04743.pdf

Future of INFERNO

Applications

- Application of INFERNO to a real world CMS analysis
- Optimizations of detector designs
- Development of a common package across experiments

Computational needs

- Training and optimization needs powerful GPUs
- Data preprocessing and ML will benefit from Spark / Kubernetes clusters
- Ideally move entirely to cloud computing

Summary

- INFERNO shows promising performance on toy problems

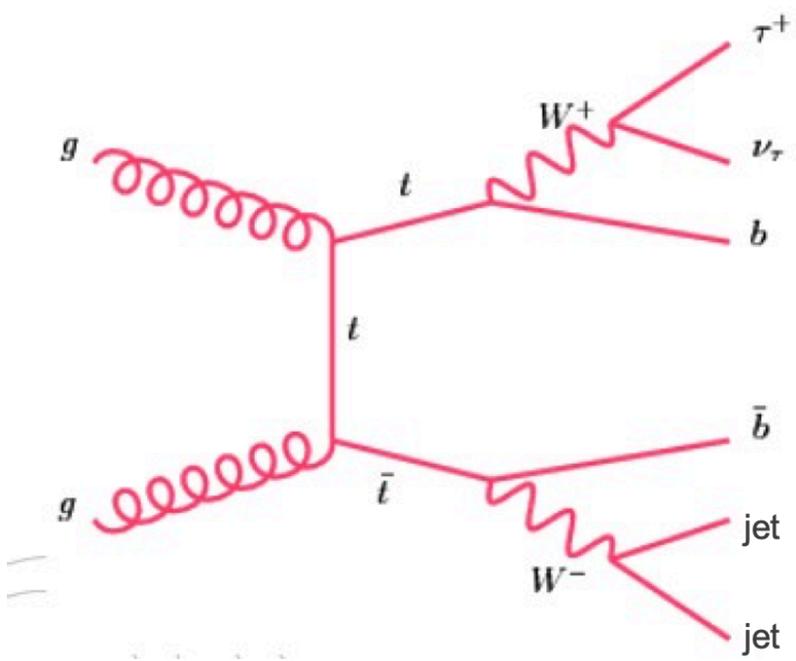
- Several advanced applications of INFERNO are planned / have started

- Need for powerful hardware

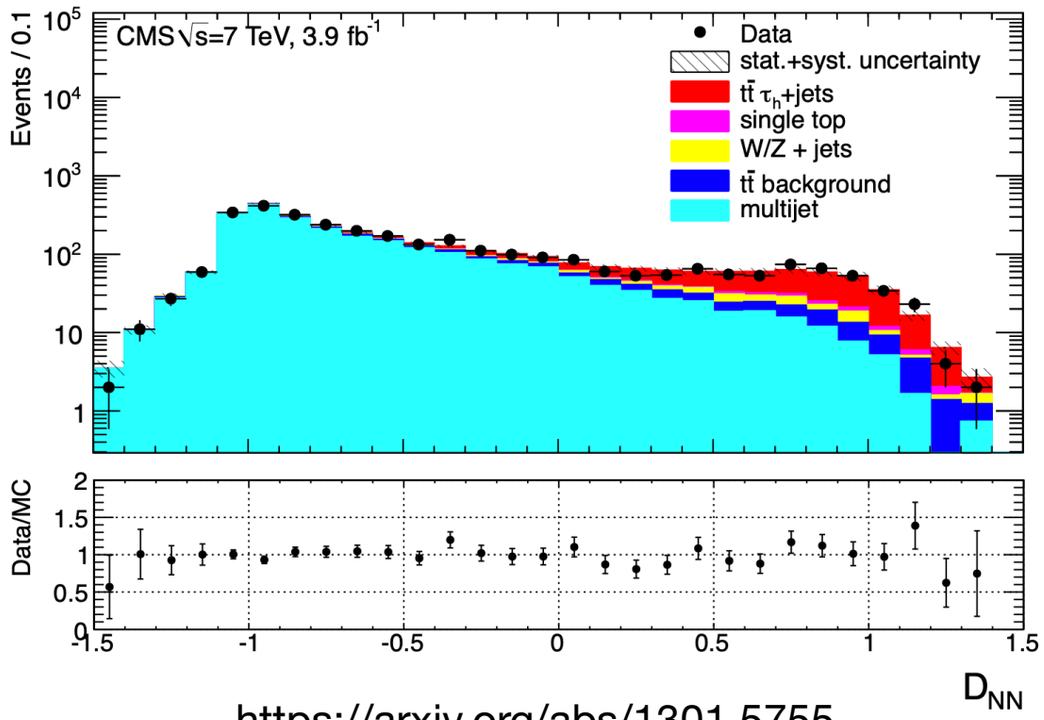
BACKUP

Top pair cross section with CMS open data

TOP-11-004 - Top pair cross section in tau+jets



σ(tt) = 152 ± 12 (stat.) ± 32 (syst.) ± 3 (lum.) pb



https://arxiv.org/abs/1301.5755

- Replication of full TOP-11-004 with CMS open data including systematics
- Relevant 2011 legacy samples available for Data and MC
- Analysis dominated by systematic uncertainties and uses a ANN with TMVA
- Simple analysis, but can be made more challenging by analyzing more decay channels

Status: implemented basic framework to analyze CMS open data with full systematic uncertainties

Recent related work on differentiable analysis

Work by S. Wunsch et. al (CMS)

- **Optimal statistical inference** in the presence of **systematic uncertainties** using neural network optimization based on **binned Poisson likelihoods** with **nuisance parameters** (https://arxiv.org/abs/2003.07186)

- **Idea**: construct binned Poisson Likelihood by using **histograms** and **approximate the gradient** of the bin function by the **derivative of a Gaussian** → avoids the softmax approximation

Work by N. Simpson, L. Heinrich et. al (ATLAS)

- **Idea**: optimize neural networks with **differentiable profile likelihood** in the loss function via **fixed-point differentiation** → allows to directly **optimize CLs limits**

- Library ***neos*** based on autodiff library **JAX** and library **FAX** for differentiating fixed point problems in JAX

https://github.com/pyhf/neos

HSF activity for differentiable analysis

- New **HSF activity** for differential analysis related to various HEP problems **across experiments**

- Discussion channel for the community on gitter