Accelerating HEP theory with ML models

Stefano Carrazza

22-26 June 2020, MCnet Machine Learning School

Università degli Studi di Milano (UNIMI and INFN Milan)

Acknowledgement: This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement no. 740006.

N3PDF

Machine Learning • PDFs • QCD

INFN

Istituto Nazionale di Fisica Nucleare

Why talk about machine learning?

Why talk about machine learning?

because

- it is an essential set of algorithms for building models in science,
- fast development of new tools and algorithms in the past years,
- nowadays it is a requirement in experimental and theoretical physics,
- large interest from the HEP community: *IML, conferences, grants*.

When apply machine learning in theoretical physics?

When apply machine learning in theoretical physics?

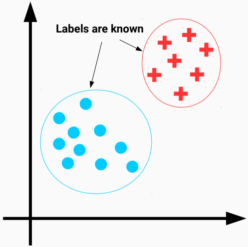
when:

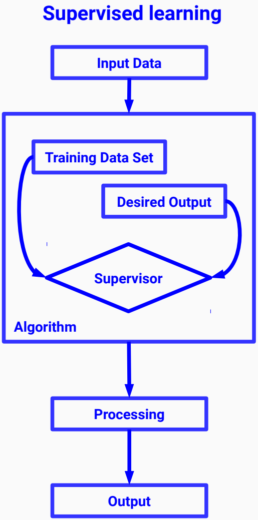
- Ambiguous choices.
- Lack of information.
- Interpolation, sampling.
- Performance acceleration.

Machine learning algorithms

Machine learning algorithms:

- Supervised learning:
  regression, classification, ...

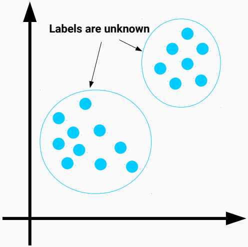


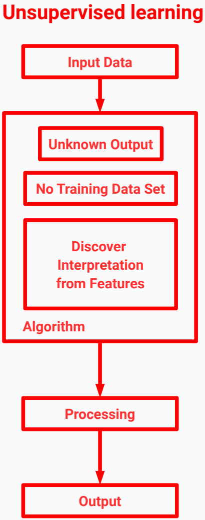


Machine learning algorithms

Machine learning algorithms:

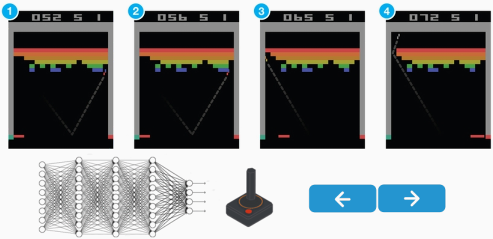
- Supervised learning:
  regression, classification, ...
- Unsupervised learning:
  clustering, dim-reduction, ...





Machine learning algorithms:

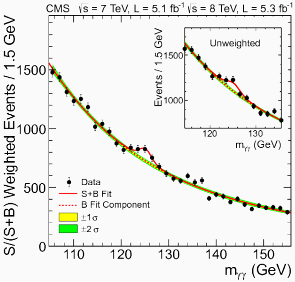
- Supervised learning:
  regression, classification, ...
- Unsupervised learning:
  clustering, dim-reduction, ...
- Reinforcement learning:
  real-time decisions, ...

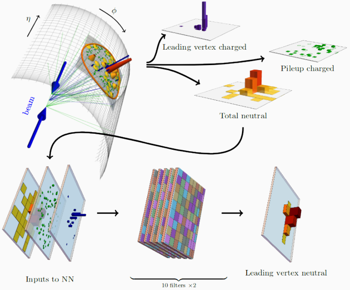


ML in HEP

Some remarkable examples are:

- **Signal-background detection:**
  Decision trees, artificial neural networks, support vector machines.
- **Jet discrimination:**
  Deep learning imaging techniques via convolutional neural networks.
- **HEP detector simulation:**
  Generative adversarial networks, *e.g.* LAGAN and CaloGAN.





Some examples of ML in HEP theory

Supervised learning:

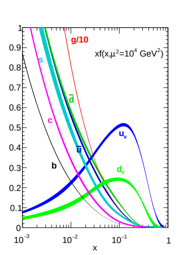
- The structure of the proton at the LHC*
- Theoretical prediction and combination
- Monte Carlo reweighting techniques*
- BSM searches and exclusion limits
- Generative models (GANs)*

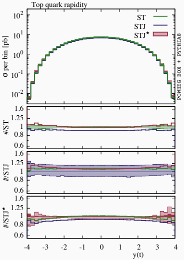
Unsupervised learning:

- Clustering and compression
- Density estimation and anomaly detection
- Monte Carlo integration*

Reinforcement learning:

- Jet grooming*

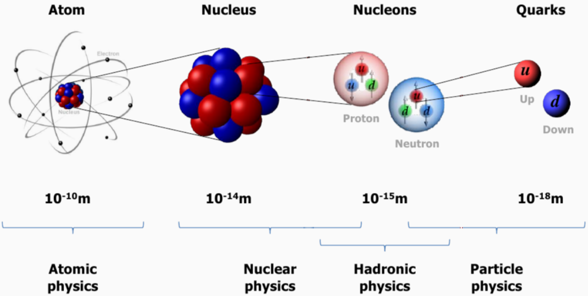




ML and Parton Density functions

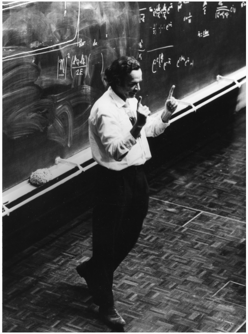
Parton density functions

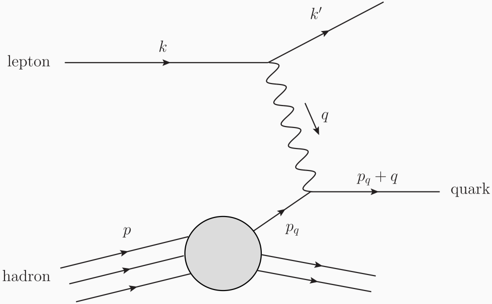
The **parton** model was introduced by Feynman in 1969 in order to characterize **hadrons** (*e.g.* protons and neutrons) in QCD processes and interactions in high energy particle collisions.



Partons are quarks and gluons characterized by a probability density functions of its nucleon momentum.

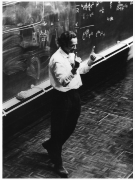
The Feynman Parton Model





- **Photon probes the proton** by striking a **free massless "parton"** (quark, gluon) that carries a fraction x of its parent proton.
- Value of x is fixed by final-state kinematics.
- Cross-section proportional to probability $q_i(x)$ of finding parton of species i with momentum-fraction x in target proton.

Perturbative QCD



- The **Parton Model** is the first order of a perturbative expansion
- **PDFs** are **not calculable**: reflect non-perturbative physics of confinement.
- **PDFs** are **essential** for a **realistic computation** of any particle physics **observable**, σ, thanks to the factorization theorem

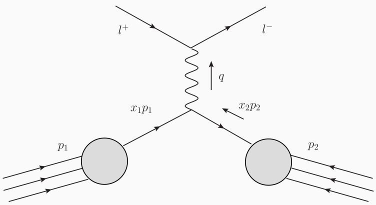
$$\sigma = \hat{\sigma} \otimes f,$$

  where the elementary **hard cross-section** $\hat{\sigma}$ is convoluted with f the **PDF**.
- Can be **proven rigorously** using the **OPE** (Wilson expansion).

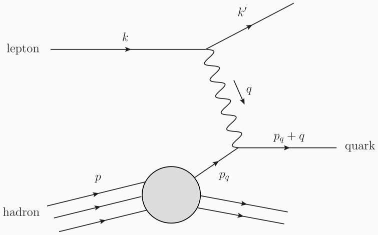
Perturbative QCD

Factorization theorem is applied to several processes:

Drell-Yan (e.g. LHC)

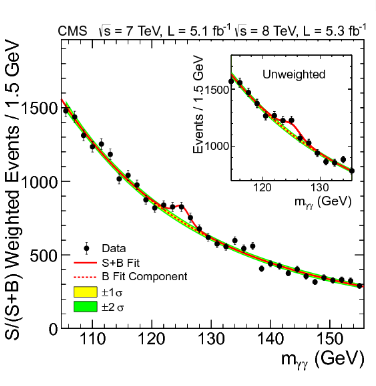


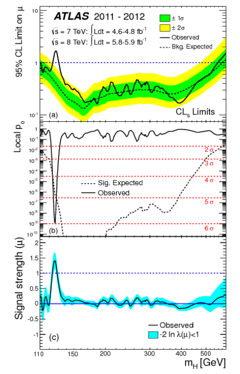
DIS



PDFs are **extracted** by comparing theoretical predictions to real data.

- PDFs are **necessary** to determine theoretical predictions for **signal**/**background** of experimental **measurements**.
  - *e.g. the Higgs discovery at the LHC*:

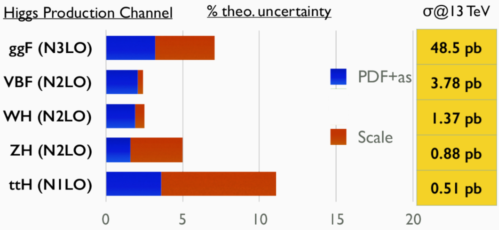




PDF uncertainties

PDF determination requires a sensible estimate of the **uncertainty**, and not only the central value, so not a well researched topic in ML.

CERN Yellow Report 4 (2016)



PDF uncertainties are a **limiting** factor in the accuracy of theoretical predictions for several processes at LHC.

⇒ Need of **precise** PDF determination and **uncertainty** estimate.

Parton density functions

Historical examples of the first PDF models:

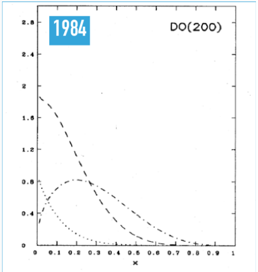
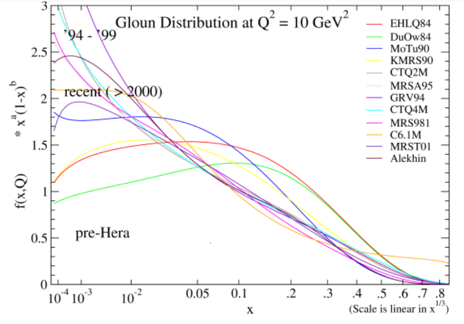


FIG. 27. "Soft-gluon" ($\Lambda = 200$ MeV) parton distributions of Duke and Owens (1984) at $Q^2 = 5$ GeV^2: valence quark distribution $x[u_v(x)+d_v(x)]$ (dotted-dashed line), $xG(x)$ (dashed line), and $q_v(x)$ (dotted line).

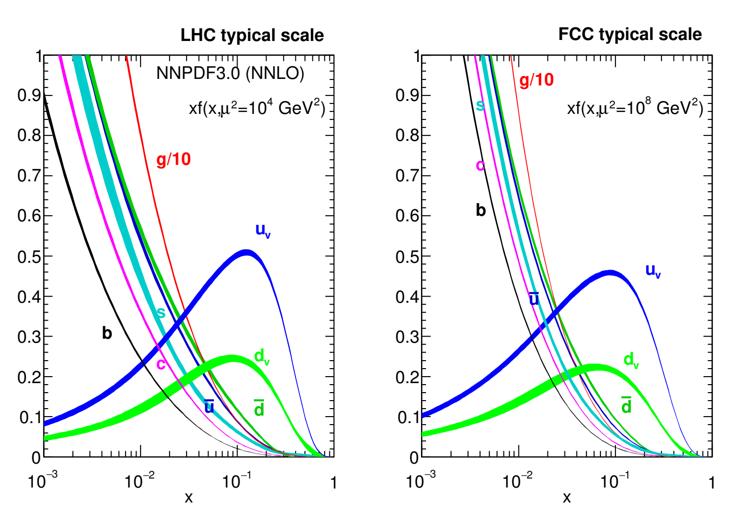


where

- PDFs are very simple functional forms (polynomials).
- PDFs are constrained by few data points and low order theory.
- No uncertainties are provided.
- No cross-validation methods are implemented.

Parton density functions

Possible improvement: use ML in PDF determination.
NNPDF (Neural Network PDFs) created **10 years ago**.



Why ML in PDFs determination?

- **PDFs** are **essential** for a **realistic computation** of hadronic particle physics **observable**, σ, thanks to the factorization theorem, *e.g.* in pp collider:

$$\underbrace{\sigma_X(s, M_X^2)}_{Y} = \sum_{a,b} \int_{x_{\min}}^{1} dx_1 dx_2 \, \underbrace{\hat{\sigma}_{a,b}(x_1, x_2, s, M_X^2)}_{X} \, f_a(x_1, M_X^2) f_b(x_2, M_X^2),$$

  where the elementary **hard cross-section** $\hat{\sigma}$ is convoluted with f the **PDF**.
- $f_i(x_1, M_X^2)$ is the PDF of parton i carrying a fraction of momentum x at scale $M \Rightarrow$ **needs to be learned from data.**

Why ML in PDFs determination?

- **PDFs** are **essential** for a **realistic computation** of hadronic particle physics **observable**, σ, thanks to the factorization theorem, *e.g.* in pp collider:

$$\underbrace{\sigma_X(s, M_X^2)}_{Y} = \sum_{a,b} \int_{x_{\min}}^{1} dx_1 dx_2 \, \underbrace{\hat{\sigma}_{a,b}(x_1, x_2, s, M_X^2)}_{X} \, f_a(x_1, M_X^2) f_b(x_2, M_X^2),$$

  where the elementary **hard cross-section** $\hat{\sigma}$ is convoluted with f the **PDF**.

- $f_i(x_1, M_X^2)$ is the PDF of parton i carrying a fraction of momentum x at scale M $\Rightarrow$ **needs to be learned from data.**

- Constraints come in the form of convolutions:

$$X \otimes f \to Y$$

- Experimental data points is ˜5000 $\to$ not a big data problem
- Data from several process and experiments over the past decades $\Rightarrow$ deal with **data inconsistencies**

The NNPDF methodology

The NNPDF (Neural Networks PDF) implements the Monte Carlo approach to the determination of a global PDF fit. We propose to:

1. **reduce** all sources of **theoretical bias**:
    - no fixed functional form
    - possibility to reproduce non-Gaussian behavior

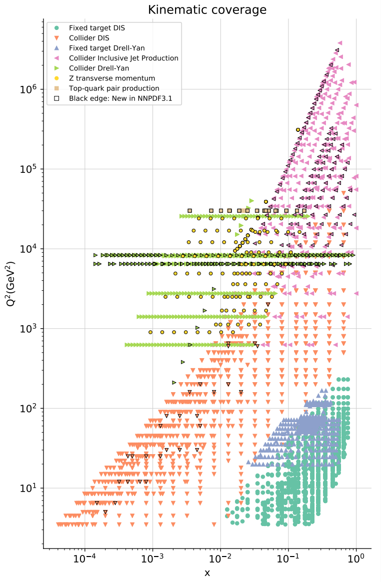
   $\Rightarrow$ use Neural Networks instead of polynomials
2. provide a sensible estimate of the **uncertainty**:
    - uncertainties from input experimental data
    - minimization inefficiencies and degenerate minima
    - theoretical uncertainties

   $\Rightarrow$ use MC artificial replicas from data, training with a GA minimizer
3. Test the setup through **closure tests**

Experimental data

The total number of data points for the default PDF determination is

- 4175 at LO, 4295 at NLO and 4285 at NNLO.
- 7 physical processes from 14 experiments over ~30 years (deal with data inconsistencies)
- few data points at high and low x (deal with extrapolation)
- range of 5 and 7 orders of magnitude per PDF evaluation arguments (x, Q^2)



DGLAP evolution

Can we reduce the PDF input size? **Yes**, thanks to DGLAP:

$$f_i(x_\alpha, Q^2) = \Gamma(Q, Q_0)_{ij\alpha\beta} f_j(x_\beta, Q_0^2)$$

We remove the Q^2 dependence from PDF determination thanks to the DGLAP evolution operator Γ.

$$f(x, Q^2) \to f(x, Q_0^2) := f(x)$$

- Precompute the DGLAP operator for all data points
- Apply the operator to the partonic cross section
- Store the results and perform fast convolutions

In NNPDF theoretical predictions are stored in **APFELgrid** tables:

$$\sigma = \sum_{i,j}^{n_f} \sum_{\alpha,\beta}^{n_x} W_{ij\alpha\beta} f_i(x_\alpha, Q_0^2) f_j(x_\beta, Q_0^2)$$

Defining the ML problem

In comparison to a typical ML problem, a PDF fit

- requires a statistically sound uncertainty estimate
- is a regression problem but complex dependence on PDFs
- must satisfy physical constrains:
  - $f(x) \to 0$ for $x \to 1$ (continuity)
  - sum rules:

$$\sum_i^{n_f} \int_0^1 dx\, x f_i(x) = 1, \qquad \int_0^1 dx\, (u(x) - \bar{u}(x)) = 2$$

$$\int_0^1 dx\, (d(x) - \bar{d}(x)) = 1, \quad \int dx\, (q(x) - \bar{q}(x)) = 0,\ q = s, b, t$$

PDF parametrizations

- Early models:

$$f_i(x) = A \cdot x^{\alpha}(1-x)^{\beta}$$

  - parameters are chosen based on Hessian minimization approach
  - Can a simple model provide a reliable uncertainty estimate?
  - Can it deal with data inconsistencies?

PDF parametrizations

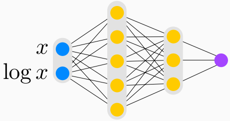
- Early models:

$$f_i(x) = A \cdot x^{\alpha}(1-x)^{\beta}$$

  - parameters are chosen based on Hessian minimization approach
  - Can a simple model provide a reliable uncertainty estimate?
  - Can it deal with data inconsistencies?

- NNPDF approach:

$$f_i(x, Q_0) = A \cdot x^{\alpha}(1-x)^{\beta} NN(x)$$



  - fully connected MLP (2-5-3-1)
  - two sigmoid hidden layers and linear output layer
  - x8 independent PDFs $\Rightarrow$ 296 free parameters

Cost function

- We minimize the cost function:

$$\chi^2 = \sum_{ij} (D_i - O_i)\sigma^{-1}_{i,j}(D_j - O_j)$$

  - D_i is the experimental measurement for point i
  - O_i the theoretical prediction for point i $(= \bar{\sigma} \otimes f)$
  - σ_{ij} is the covariance matrix between points i and j with corrections for normalization uncertainties
  - supplemented by additional penalty terms for positivity observables

Propagating experimental uncertainties

Generate artificial **Monte Carlo** data replicas from experimental data.

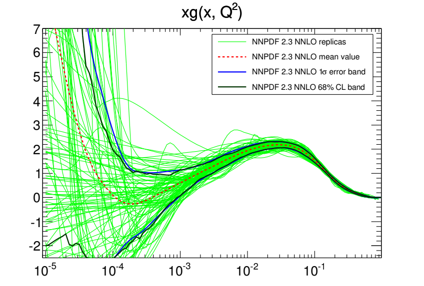
We perform N_{rep} O(1000) fits, sampling pseudodata replicas:

$$D_i^{(r)} \to D_i^{(r)} + \mathrm{chol}(\Sigma)_{i,j}\mathcal{N}(0,1), \quad i,j = 1..N_{\mathrm{dat}},\, r = 1...N_{\mathrm{rep}}$$

We obtain N_{rep} PDF replicas. No assumptions at all about the Gaussianity of the errors.

PDF fit example

The procedure delivers a **Monte Carlo** representation of results:



The central value of observables based on PDFs are obtained with:

$$\langle \mathcal{O}[f] \rangle = \frac{1}{N_{\rm rep}} \sum_{k=1}^{N_{\rm rep}} \mathcal{O}[f_k]$$

Optimization algorithm

The current approach is genetic optimization, based on nodal mutation probabilities and more recently the covariance matrix evolution strategy

$$w \to w + \eta \frac{r_\delta}{N_{\text{ite}}^{r_{\text{ite}}}}, \quad \eta = 15,\, r_\delta \sim U(-1,1),\, r_{\text{ite}} \sim U(1,0)$$

At each iteration, generate 80 mutants and select best mutant.

Advantages

- Simple to implement and understand.
- Good dealing with complex analytic behavior.
- Doesn't require evaluating the gradient.

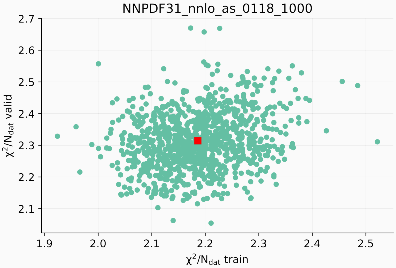
Disadvantages

- May not be close to a global minimum.
- Requires many functions evaluations.
- Needs tuning.

Stopping

We have cross-validation implemented:

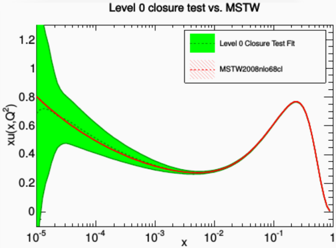
- We split data in a training and validation set.
- Training fraction is 50%, different for each replica.
- We perform the GA on the training set for a fixed number of iterations O(30000).
- Stop at the minimum of the validation set, storing the parameters from the replica at that iteration.



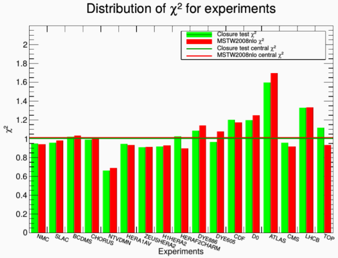
Validation with closure test

Closure tests

- Assume that the underlying PDF is known, generate data, fluctuations around the prediction of the true PDF.
- Perform a fit and compare to underlying PDF.
- Check that the results are consistent.



(a) Level 0 fit

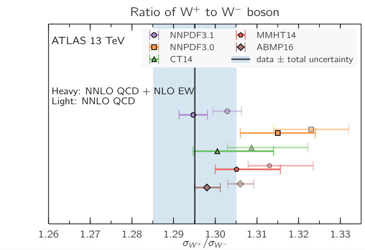


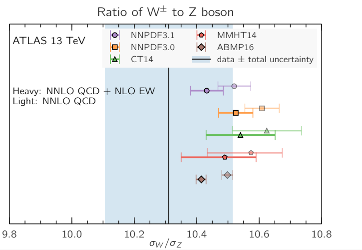
(b) Level 2 results

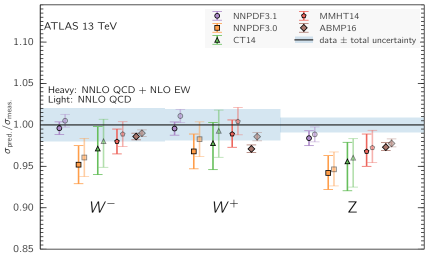
Level 0: Fit predictions of the true PDF without fluctuations. $\chi^2/N_{\text{dat}} \to 0$.

Level 2: Generate pseudodata replicas on top of replicas. $\chi^2/N_{\text{dat}} \to 1$.

W and Z production cross-sections at LHC 13 TeV

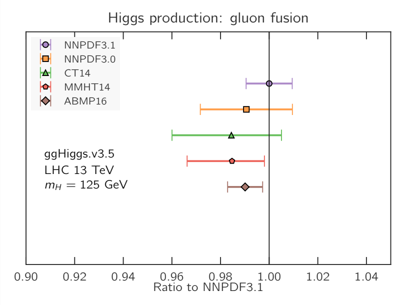


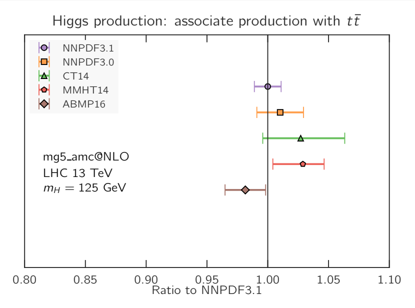


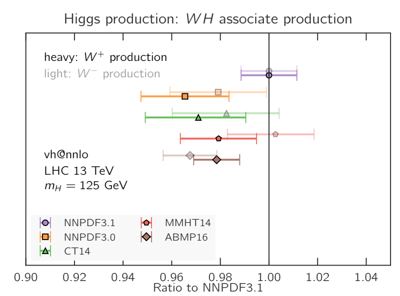


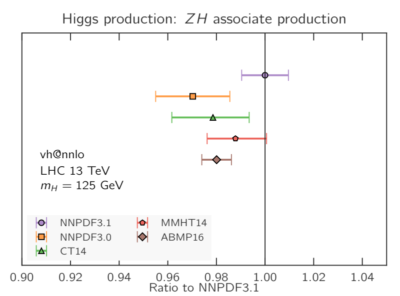
NNPDF3.1 have smaller PDF uncertainties than NNPDF3.0.

Higgs production cross-sections









Towards deep learning PDFs

PDF challenges

Challenges:

- How to increase **fit performance speed**?
  - faster fits $\Rightarrow$ more fits
- How can we **tune/learn the methodology**?
  - select the best model for our data/theory

PDF challenges

Challenges:

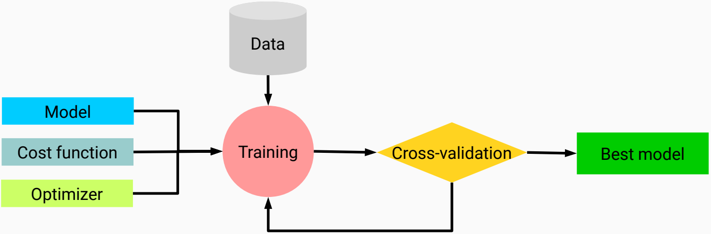
- How to increase **fit performance speed**?
  - faster fits $\Rightarrow$ more fits
- How can we **tune/learn the methodology**?
  - select the best model for our data/theory

Solution $\Rightarrow$ move towards **deep learning**

- in terms **software/technology**
- in terms of **methodology**

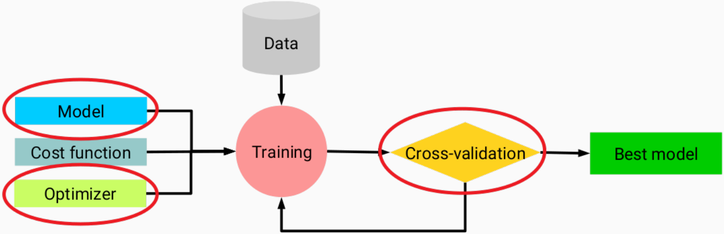
Deep learning pipeline

PDF determination is a supervised learning problem thus we need to provide review for the following sectors:

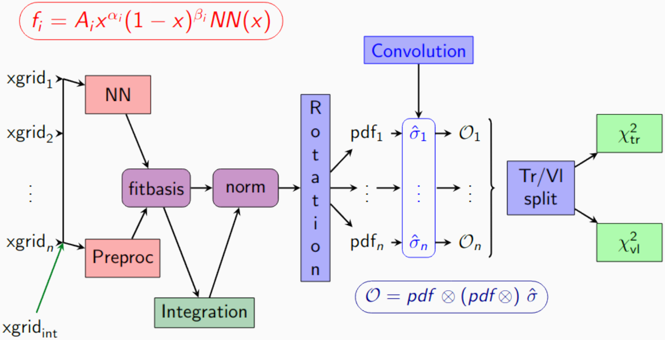


Deep learning pipeline

PDF determination is a supervised learning problem thus we need to provide review for the following sectors:



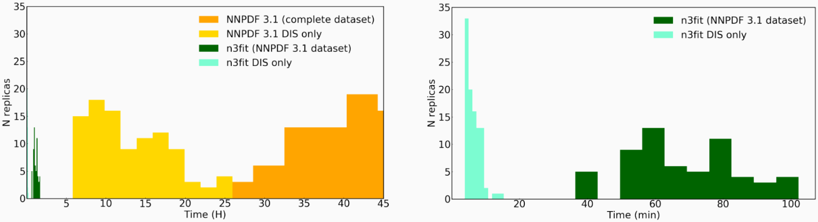
The `n3fit` model



New features:

- `Python/C++` implementation using **TensorFlow**
- Modular approach $\Rightarrow$ easier and faster development
- Can vary all aspects of the methodology

Performance benefits - time per replica



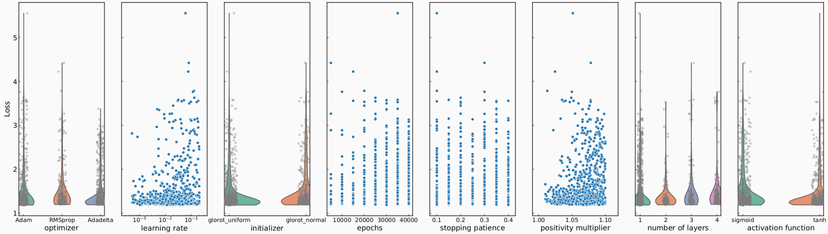
Benefits

- Gain on speed and efficiency, less **CPU hours for a fit**
- Usage of new technologies $\rightarrow$ hardware, libraries
- Usage of **gradient descent** optimization methods

$\Rightarrow$ **Possibility to learn and tune the methodology**

Learning the methodology

How to determine the best methodology?



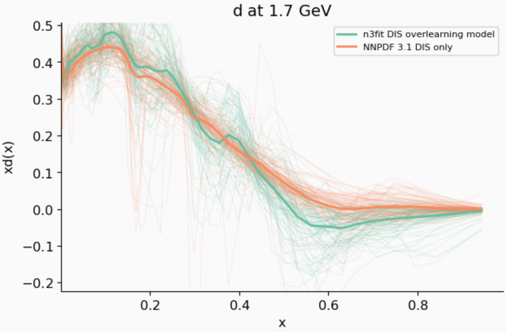
Perform **hyperoptimization scans**:

Neural Network	Fit options
Number of layers (*)	Optimizer (*)
Size of each layer	Initial learning rate (*)
Dropout	Maximum number of epochs (*)
Activation functions (*)	Stopping Patience (*)
Initialization functions (*)	Positivity multiplier (*)

- Optimize figure of merit: validation χ^2
- Use bayesian updating (`hyperopt`)

The overfitting problem

Using validation set χ^2:

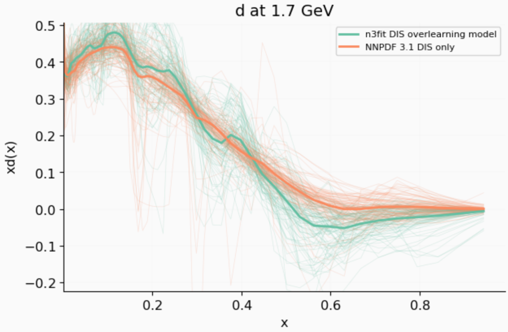


The choice of the **right figure of merit** is important:

- **NNPDF** wiggles $\rightarrow$ finite size , goes away as N_{rep} grows
- **N3PDF** wiggles $\rightarrow$ **overfitting**, correlations training-validation data!

The overfitting problem

Using validation set χ^2:



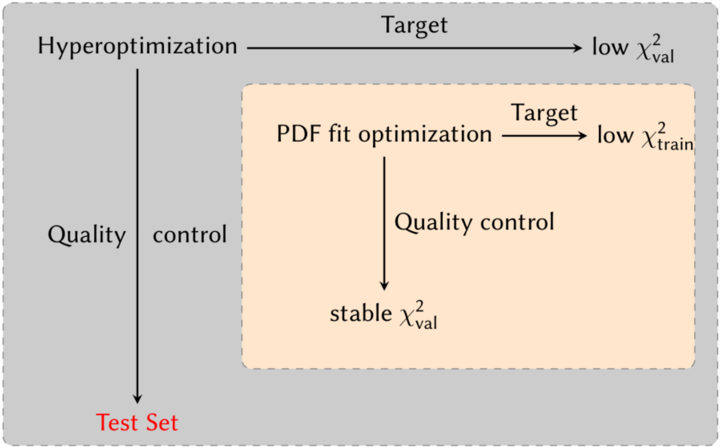
The choice of the **right figure of merit** is important:

- **NNPDF** wiggles $\rightarrow$ finite size , goes away as N_{rep} grows
- **N3PDF** wiggles $\rightarrow$ **overfitting**, correlations training-validation data!

$\Rightarrow$ **define a proper quality control criterion**

Cross-Validation vs hyperoptimization

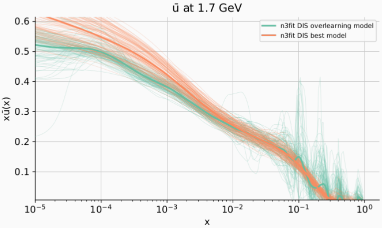
Define a completely uncorrelated **Test Set**

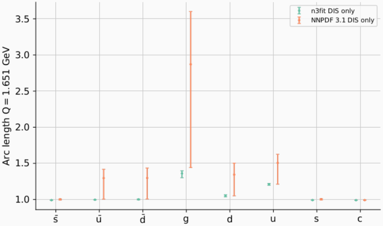


Optimize on weighted average of validation and test.

Removing overfitting

Using test-validation set χ^2:





- No **overfitting**
- Greater **stability**
- Reduced uncertainties

	DIS only	Global
`n3fit` (new)	1.10	1.15
`nnfit` (old)	1.13	1.16

Quality control

Chronological fits

Idea:

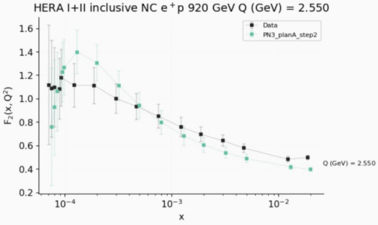
1. Take a pre-HERA dataset
2. Perform hyperoptimization
3. Compare predictions to “future” data

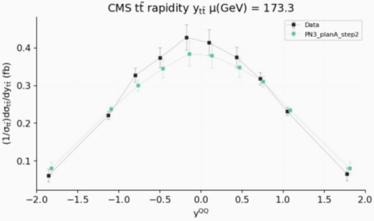
Chronological fits

Idea:

1. Take a pre-HERA dataset
2. Perform hyperoptimization
3. Compare predictions to “future” data

Examples:





⇒ Results within PDF uncertainty!

Defining a proper Test set

How to define a proper Test Set?

- we have a limited dataset with lots of features, $N_{\mathrm{data}} \approx 5000$

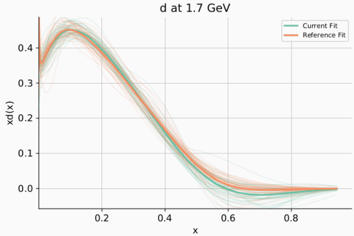
Defining a proper Test set

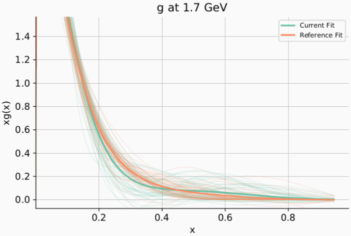
How to define a proper Test Set?

- we have a limited dataset with lots of features, $N_{\mathrm{data}} \approx 5000$

$\Rightarrow$ **Potential solution:** use k-fold cross-validation.

- Use k partitions in a rotation estimation for the Test Set
- hyperoptimize the mean value of the Test Set χ^2





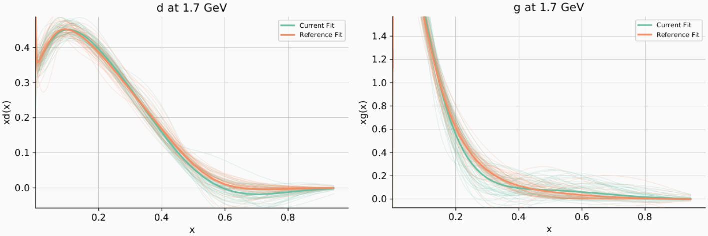
Defining a proper Test set

How to define a proper Test Set?

- we have a limited dataset with lots of features, $N_{\text{data}} \approx 5000$

$\Rightarrow$ **Potential solution:** use k-fold cross-validation.

- Use k partitions in a rotation estimation for the Test Set
- hyperoptimize the mean value of the Test Set χ^2



$\Rightarrow$ **Compatible with our previous Test Set definition.**

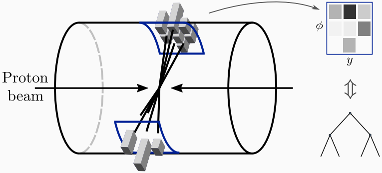
Questions?

ML and jet substructure

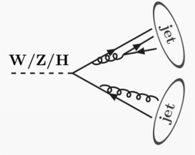
Boosted jets at the LHC

High energy collisions at the LHC $\Rightarrow$ **boosted objects**:

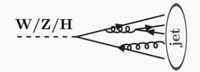
- particles such as W, Z, H, $t, \ldots$ are produced with $p_T^{\text{jet}} \gg m_{\text{jet}}$,
- hadronic collimated decays often reconstructed into single jets.



$p_T^{\text{jet}} \leq m_{\text{jet}}$



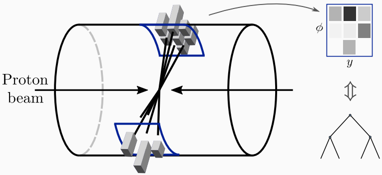
$p_T^{\text{jet}} \gg m_{\text{jet}}$



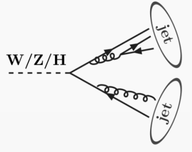
Boosted jets at the LHC

High energy collisions at the LHC $\Rightarrow$ **boosted objects**:

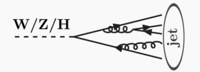
- particles such as W, Z, H, t, $\ldots$ are produced with $p_T^{\text{jet}} \gg m_{\text{jet}}$,
- hadronic collimated decays often reconstructed into single jets.



$p_T^{\text{jet}} \leq m_{\text{jet}}$



$p_T^{\text{jet}} \gg m_{\text{jet}}$

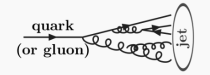


Problem: identify **hard structure** of a jet from **radiation patterns**.

(Jet from W, Z, H, t or QCD?)

$p_T^{\text{jet}} \gg m_{\text{jet}}$

?



Jet drawings by G. Soyez

Jet grooming techniques

How to identify hard structure of a jet?

- Look at the mass of the jet.
- Remove distortion due to QCD radiation and pileup.

Grooming goal $\Rightarrow$ remove unassociated soft wide-angle radiation.

Jet grooming techniques

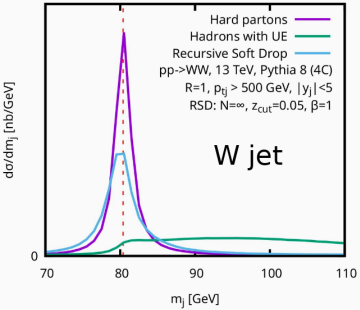
How to identify hard structure of a jet?

- Look at the mass of the jet.
- Remove distortion due to QCD radiation and pileup.

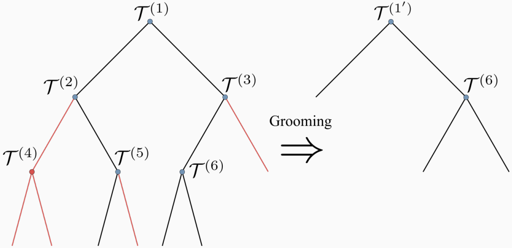
Grooming goal $\Rightarrow$ remove unassociated soft wide-angle radiation.

Jet grooming algorithms:

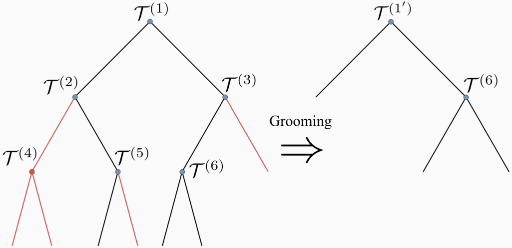
- modified MassDrop Tagger
  Dasgupta *et al.*, arXiv:1307.0007
- Soft Drop (SD)
  Larkoski *et al.*, arXiv:1402.2657
- Recursive Soft Drop (RSD)
  Dreyer *et al.*, arXiv:1804.03657



(Recursive) Soft Drop algorithm



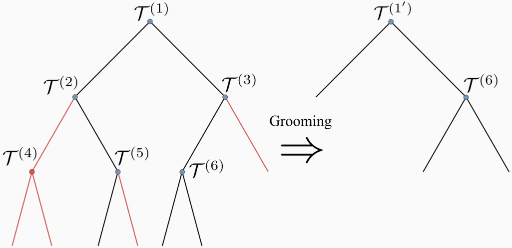
1. Cast jet as clustering tree with nodes $\mathcal{T}^{(i)}$ and children nodes a, b.



1. Cast jet as clustering tree with nodes $\mathcal{T}^{(i)}$ and children nodes a, b.
2. Define state of each node as $s_t = \{z, \Delta_{ab}\}$ where

$$z = \frac{\min(p_{t,a}, p_{t,b})}{p_{t,a} + p_{t,b}}, \quad \Delta_{ab}^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2$$

(Recursive) Soft Drop algorithm



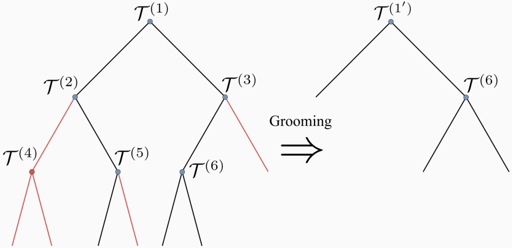
1. Cast jet as clustering tree with nodes $\mathcal{T}^{(i)}$ and children nodes a, b.
2. Define state of each node as $s_t = \{z, \Delta_{ab}\}$ where

$$z = \frac{\min(p_{t,a}, p_{t,b})}{p_{t,a} + p_{t,b}}, \quad \Delta_{ab}^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2$$

3. Evaluate policy (β, z_{cut} and R_0 are free parameters):

$$\pi_{\mathrm{RSD}}(s_t) = \begin{cases} 0 & \text{if} \quad z > z_{\mathrm{cut}} \left(\frac{\Delta_{ab}}{R_0}\right)^{\beta} \\ 1 & \text{else} \end{cases}$$

(Recursive) Soft Drop algorithm



1. Cast jet as clustering tree with nodes $\mathcal{T}^{(i)}$ and children nodes a, b.
2. Define state of each node as $s_t = \{z, \Delta_{ab}\}$ where

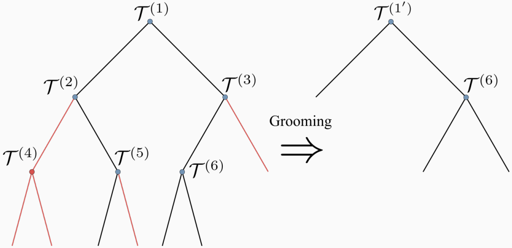
$$z = \frac{\min(p_{t,a}, p_{t,b})}{p_{t,a} + p_{t,b}}, \quad \Delta_{ab}^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2$$

3. Evaluate policy (β, z_{cut} and R_0 are free parameters):

$$\pi_{\mathrm{RSD}}(s_t) = \begin{cases} 0 & \text{if} \quad z > z_{\mathrm{cut}} \left(\frac{\Delta_{ab}}{R_0}\right)^{\beta} \\ 1 & \text{else} \end{cases}$$

4. If $\pi_{\mathrm{RSD}}(s_t) = 1 \rightarrow$ remove softer branch and update jet tree,

(Recursive) Soft Drop algorithm



1. Cast jet as clustering tree with nodes $\mathcal{T}^{(i)}$ and children nodes a, b.
2. Define state of each node as $s_t = \{z, \Delta_{ab}\}$ where

$$z = \frac{\min(p_{t,a}, p_{t,b})}{p_{t,a} + p_{t,b}}, \quad \Delta_{ab}^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2$$

3. Evaluate policy (β, z_{cut} and R_0 are free parameters):

$$\pi_{\mathrm{RSD}}(s_t) = \begin{cases} 0 & \text{if} \quad z > z_{\mathrm{cut}} \left(\frac{\Delta_{ab}}{R_0}\right)^{\beta} \\ 1 & \text{else} \end{cases}$$

4. If $\pi_{\mathrm{RSD}}(s_t) = 1 \rightarrow$ remove softer branch and update jet tree,
5. If $\pi_{\mathrm{RSD}}(s_t) = 0 \rightarrow$ stop recursion.

Our goal for this project

Goal of this project?

- Extend RSD jet grooming using Deep Learning techniques.

Our goal for this project

Goal of this project?

- Extend RSD jet grooming using Deep Learning techniques.

Why?

- improve m_{jet} resolution,
- verify model generalization and performance on different processes,
- provide a fast inference model.

Our goal for this project

Goal of this project?

- Extend RSD jet grooming using Deep Learning techniques.

Why?

- improve m_{jet} resolution,
- verify model generalization and performance on different processes,
- provide a fast inference model.

How?

- using Deep Reinforcement Learning (DRL) techniques.

A deep learning approach

Grooming a jet tree with DRL

Input data:

Generate jet events with Monte Carlo. Define a set of possible **states** in a five dimensional box:

$$s_t = \{z, \Delta_{ab}, \phi, m, k_t\}$$

Methodology:

Jet grooming is characterized by a policy and a sequential set of actions/cuts, so:

- Train a reinforcement learning agent which learns how to decide which action to take.
- Define an environment reward which motivates the agent to groom efficiently.

Choosing an DRL agent

Which agent?

Deep Q-Network $\rightarrow$ off-policy and discrete action space.

A deep neural network maximizes the action-value quality function:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a, \pi\right]$$

Choosing an DRL agent

Which agent?

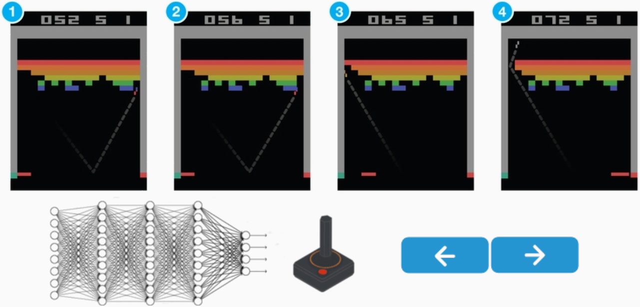
Deep Q-Network $\rightarrow$ off-policy and discrete action space.

A deep neural network maximizes the action-value quality function:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a, \pi\right]$$

A simple example:

Playing ATARI games with DRL from Minh *et al.*, arXiv:1312.5602, Nature'15:



Grooming a jet tree with DRL

DRL requirements:

- Environment definition?
  build a simulation setup where the DQN is trained and validated

Grooming a jet tree with DRL

DRL requirements:

- Environment definition?
  build a simulation setup where the DQN is trained and validated
- Reward definition?
  translate the $m_{\rm jet}$ *resolution into a game score*

Grooming a jet tree with DRL

DRL requirements:

- Environment definition?
  build a simulation setup where the DQN is trained and validated
- Reward definition?
  translate the m_{jet} *resolution into a game score*
- Hyperparameter tune?
  obtain the best model for our specific problem

Grooming a jet tree with DRL

DRL requirements:

- Environment definition?
  build a simulation setup where the DQN is trained and validated
- Reward definition?
  translate the m_{jet} resolution into a game score
- Hyperparameter tune?
  obtain the best model for our specific problem

In practice we implement the DRL framework using:

- Python $\in$ (`Keras-RL`, `TensorFlow`, `OpenAI Gym`, `hyperopt`)
- Public code available at `https://github.com/JetsGame`

Environment

Defining a jet grooming game:

Game score $\Rightarrow$ reward function *(next slides)*

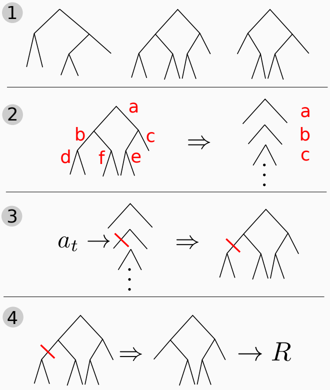
Environment

Defining a jet grooming game:

Game score $\Rightarrow$ reward function *(next slides)*

Game environment:

1. Initialize list of all trees for training.
2. Each episode starts by randomly selecting a tree and adding its root to a priority queue (ordered in Δ_{ab}).
3. Each step removes first node from priority queue, then takes action on removal of soft branch based on s_t.
4. After action, update kinematics of parent nodes, add current children to priority queue, and evaluate reward.
5. Episode terminates once priority queue is empty.



Reward function

We construct a reward function based on two components:

$$R(m, a_t, \Delta, z) = R_M(m) + \frac{1}{N_{\mathrm{SD}}} R_{\mathrm{SD}}(a_t, \Delta, z)$$

so the DQN agent is motivated to:

- improve jet mass resolution $\Rightarrow$ increase R_M,
- replicate Soft-Drop behavior $\Rightarrow$ increase R_{SD}.

Reward function

We construct a reward function based on two components:

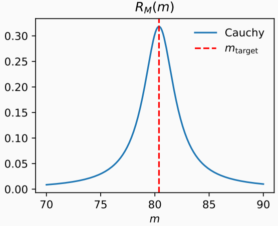
$$R(m, a_t, \Delta, z) = R_M(m) + \frac{1}{N_{\mathrm{SD}}} R_{\mathrm{SD}}(a_t, \Delta, z)$$

so the DQN agent is motivated to:

- improve jet mass resolution $\Rightarrow$ increase R_M,
- replicate Soft-Drop behavior $\Rightarrow$ increase R_{SD}.

The mass reward is defined using a Cauchy distribution:

$$R_M(m) = \frac{\Gamma^2}{\pi\left(|m - m_{\mathrm{target}}|^2 + \Gamma^2\right)}$$



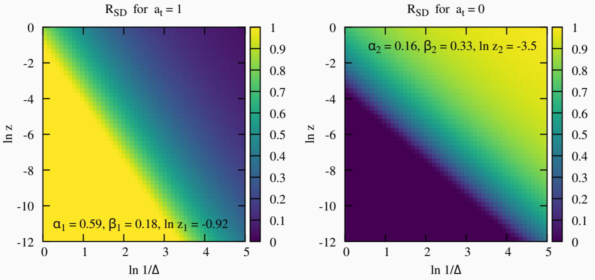
Reward function

The Soft-Drop reward is defined as

$$R_{\mathrm{SD}}(a_t, \Delta, z) = a_t \min\left(1, e^{-\alpha_1 \ln(1/\Delta)+\beta_1 \ln(z_1/z)}\right) + (1+a_t) \max\left(0, 1 - e^{-\alpha_2 \ln(1/\Delta)+\beta_2 \ln(z_2/z)}\right),$$

so the DQN agent is motivated to:

- remove wide-angle soft radiation
- keep hard-collinear emissions



Adding a multi-level approach

What about background events?

Potential mass bias for background events $\Rightarrow$ use multi-level training:

1. add to the training set signal and background samples
   $\Rightarrow$ 500k W/QCD jets simulated with Pythia 8

Adding a multi-level approach

What about background events?

Potential mass bias for background events $\Rightarrow$ use multi-level training:

1. add to the training set signal and background samples
   $\Rightarrow$ 500k W/QCD jets simulated with Pythia 8
2. at each episode randomly select a signal or background jet.
   $\Rightarrow$ adjust $R_M(m)$ accordingly to signal/background

Adding a multi-level approach

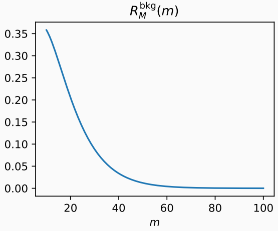
What about background events?

Potential mass bias for background events $\Rightarrow$ use multi-level training:

1. add to the training set signal and background samples
   $\Rightarrow$ 500k W/QCD jets simulated with Pythia 8
2. at each episode randomly select a signal or background jet.
   $\Rightarrow$ adjust $R_M(m)$ accordingly to signal/background

In the background case, the mass reward term is changed to:

$$R_M^{\mathrm{bkg}}(m) = \frac{m}{\Gamma_{\mathrm{bkg}}} \exp\left(-\frac{m}{\Gamma_{\mathrm{bkg}}}\right)$$



Hyperparameter tune

Free parameters to be determined:

- DQN architecture $\Rightarrow$ *(layers, nodes, activations, ...)*
- Reward parameters $\Rightarrow$ $(\alpha_{1,2},\ \beta_{1,2},\ z_{1,2},\ \Gamma)$
- Learning parameters $\Rightarrow$ *(optimizer, learning rate, ...)*

Hyperparameter tune

Free parameters to be determined:

- DQN architecture $\Rightarrow$ *(layers, nodes, activations, ...)*
- Reward parameters $\Rightarrow$ $(\alpha_{1,2},\ \beta_{1,2},\ z_{1,2},\ \Gamma)$
- Learning parameters $\Rightarrow$ *(optimizer, learning rate, ...)*

How?

Use distributed asynchronous hyperparameter optimization $\Rightarrow$ `hyperopt`.

1. Create a validation set with 50k signal (W) and background (QCD) jets.

Hyperparameter tune

Free parameters to be determined:

- DQN architecture ⇒ *(layers, nodes, activations, ...)*
- Reward parameters ⇒ ($\alpha_{1,2}$, $\beta_{1,2}$, $z_{1,2}$, Γ)
- Learning parameters ⇒ *(optimizer, learning rate, ...)*

How?

Use distributed asynchronous hyperparameter optimization ⇒ `hyperopt`.

1. Create a validation set with 50k signal (W) and background (QCD) jets.
2. Derive groomed jet mass distribution from validation set and determine:

Hyperparameter tune

Free parameters to be determined:

- DQN architecture ⇒ *(layers, nodes, activations, ...)*
- Reward parameters ⇒ $(\alpha_{1,2},\ \beta_{1,2},\ z_{1,2},\ \Gamma)$
- Learning parameters ⇒ *(optimizer, learning rate, ...)*

How?

Use distributed asynchronous hyperparameter optimization ⇒ `hyperopt`.

1. Create a validation set with 50k signal (W) and background (QCD) jets.
2. Derive groomed jet mass distribution from validation set and determine:
   - window $(w_{\min}, w_{\max})$ containing 60% of signal distribution,

Hyperparameter tune

Free parameters to be determined:

- DQN architecture ⇒ *(layers, nodes, activations, ...)*
- Reward parameters ⇒ $(\alpha_{1,2},\ \beta_{1,2},\ z_{1,2},\ \Gamma)$
- Learning parameters ⇒ *(optimizer, learning rate, ...)*

How?

Use distributed asynchronous hyperparameter optimization ⇒ `hyperopt`.

1. Create a validation set with 50k signal (W) and background (QCD) jets.
2. Derive groomed jet mass distribution from validation set and determine:
   - window $(w_{\min}, w_{\max})$ containing 60% of signal distribution,
   - the median w_{med} in that interval.

Hyperparameter tune

Free parameters to be determined:

- DQN architecture ⇒ *(layers, nodes, activations, ...)*
- Reward parameters ⇒ $(\alpha_{1,2}, \beta_{1,2}, z_{1,2}, \Gamma)$
- Learning parameters ⇒ *(optimizer, learning rate, ...)*

How?

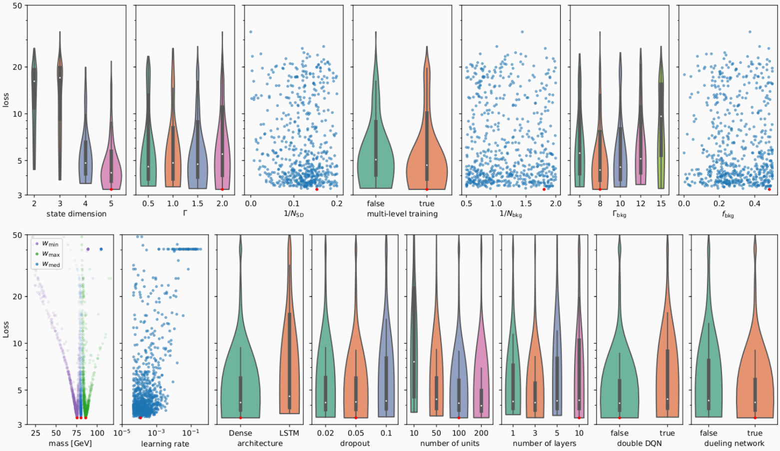
Use distributed asynchronous hyperparameter optimization ⇒ `hyperopt`.

1. Create a validation set with 50k signal (W) and background (QCD) jets.
2. Derive groomed jet mass distribution from validation set and determine:
   - window $(w_{\mathrm{min}}, w_{\mathrm{max}})$ containing 60% of signal distribution,
   - the median w_{med} in that interval.
3. Define f_{bkg} the fraction of groomed background sample $(w_{\mathrm{min}}, w_{\mathrm{max}})$:

$$\mathcal{L} = \frac{1}{5}|w_{\mathrm{max}} - w_{\mathrm{min}}| + |m_{\mathrm{target}} - w_{\mathrm{med}}| + 20 f_{\mathrm{fkg}}$$

Hyperparameter tune

Validation loss for 2000 models



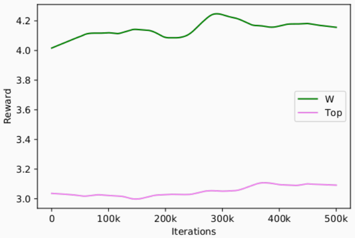
Results

Optimal GroomRL model for W and top jets

Reward evolution during the training of the GroomRL for W bosons and top quarks:

- improvement during the first 300k epochs,
- stability after 300k epochs.

$$R(m, a_t, \Delta, z)$$

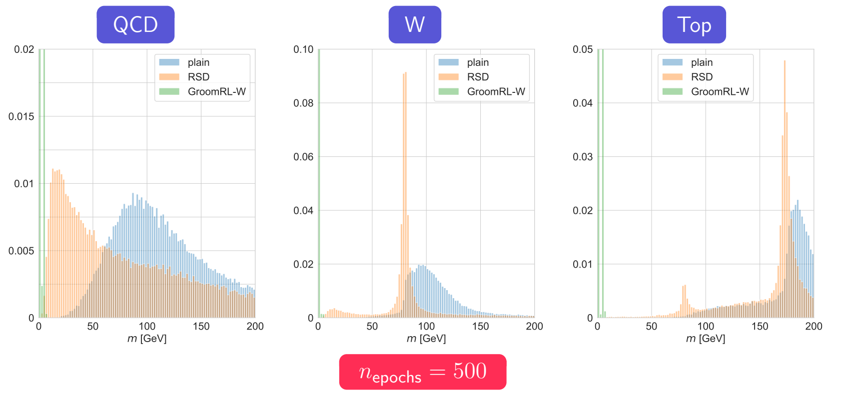


Parameters	Value
m_{target}	80.385 GeV or 173.2 GeV
s_t dimension	5
reward	Cauchy
Γ	2 GeV
$(\alpha_1, \beta_1, \ln z_1)$	$(0.59, 0.18, -0.92)$
$(\alpha_2, \beta_2, \ln z_2)$	$(0.65, 0.33, -3.53)$
$1/N_{\text{SD}}$	0.15
multi-level training	Yes
Γ_{bkg}	8 GeV
$1/N_{\text{bkg}}$	1.8 or 1.0
p_{bkg}	0.48 or 0.2
learning rate	10^{-4}
Dueling NN	Yes
Double DQN	No
Policy	Boltzmann
$N^{\text{max}}_{\text{epochs}}$	500K
Architecture	Dense
Dropout	0.05
Layers	10
Nodes	100
Optimizer	Adam

TABLE I: Final parameters for GroomRL, with the two values of m_{target} corresponding to the W and top mass.

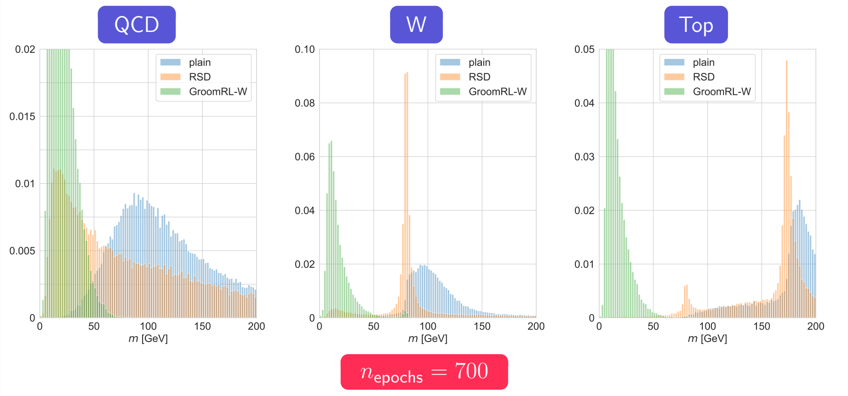
DRL training animation

`GroomRL-W` predictions vs n_{epochs}

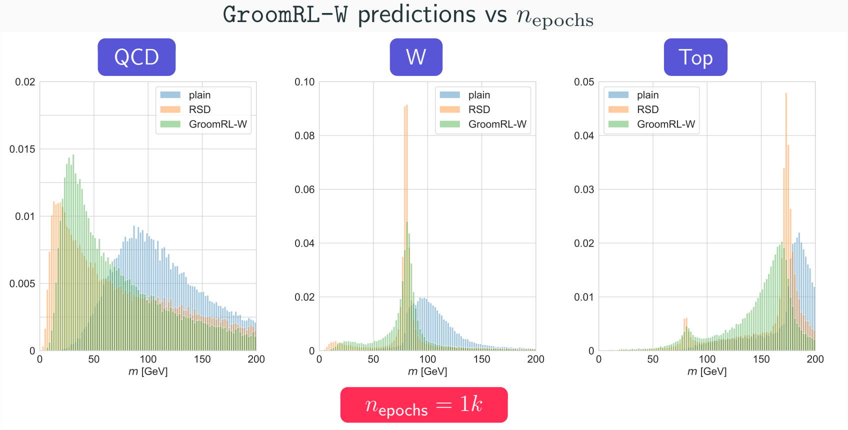


DRL training animation

GroomRL-W predictions vs n_{epochs}

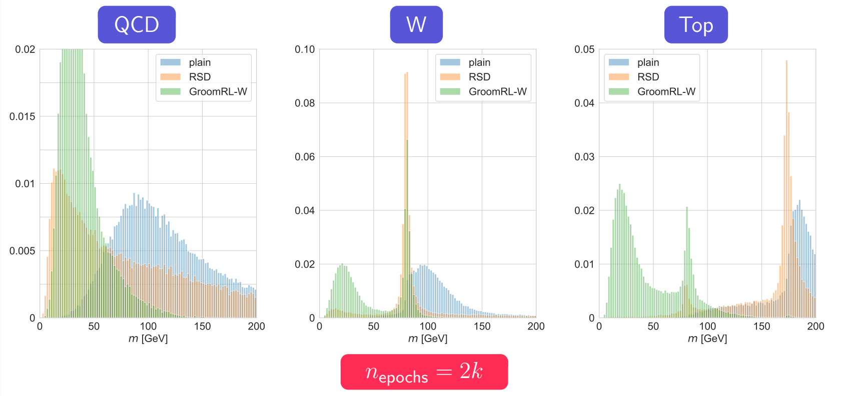


DRL training animation



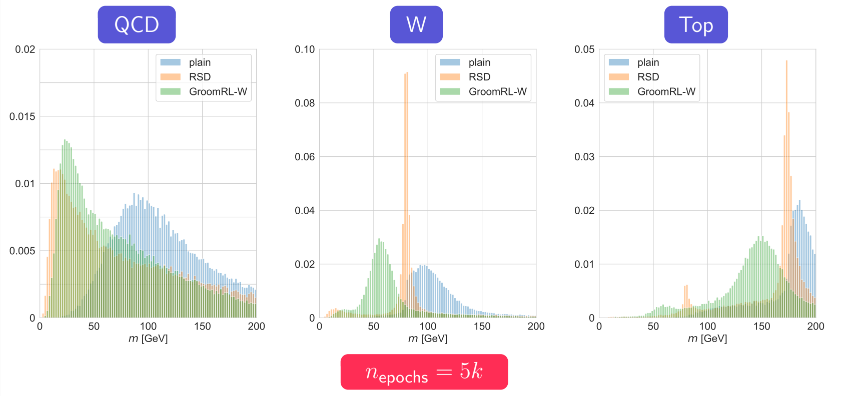
DRL training animation

GroomRL-W predictions vs n_{epochs}



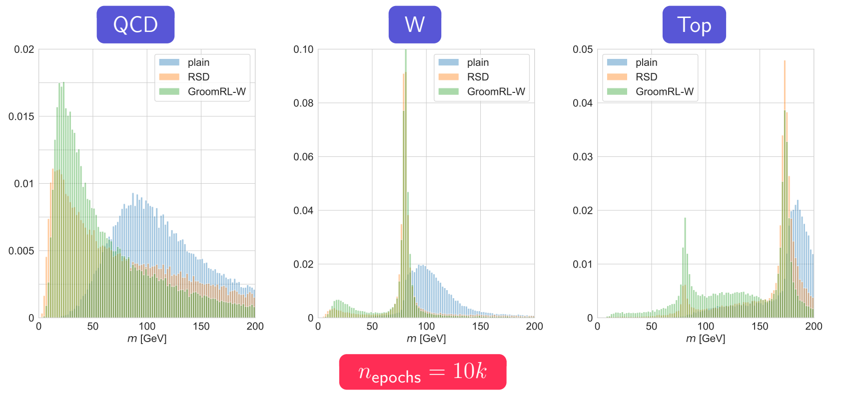
DRL training animation

$\texttt{GroomRL-W}$ predictions vs n_{epochs}



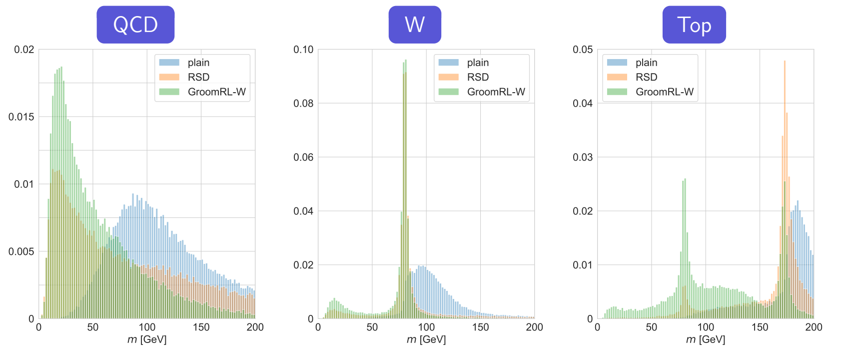
DRL training animation

GroomRL-W predictions vs n_{epochs}



DRL training animation

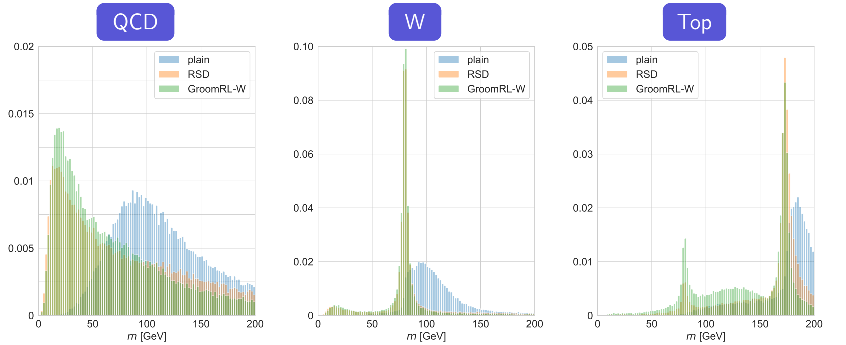
GroomRL-W predictions vs n_{epochs}



$n_{\text{epochs}} = 50k$

DRL training animation

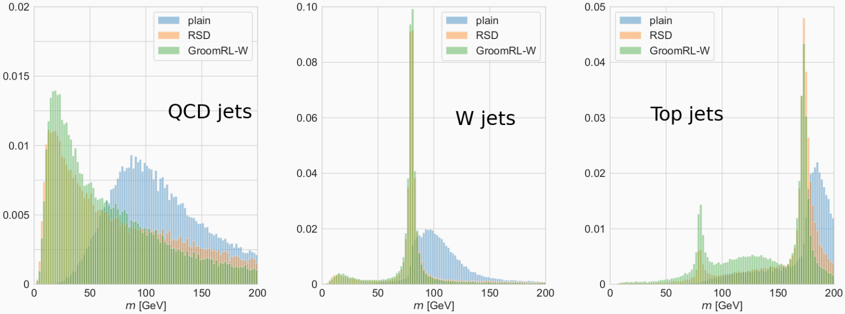
$\texttt{GroomRL-W}$ predictions vs n_{epochs}



$n_{\text{epochs}} = 500k$

Optimal `GroomRL` model for W jets

`GroomRL-W` tested on QCD, W and Top jet data

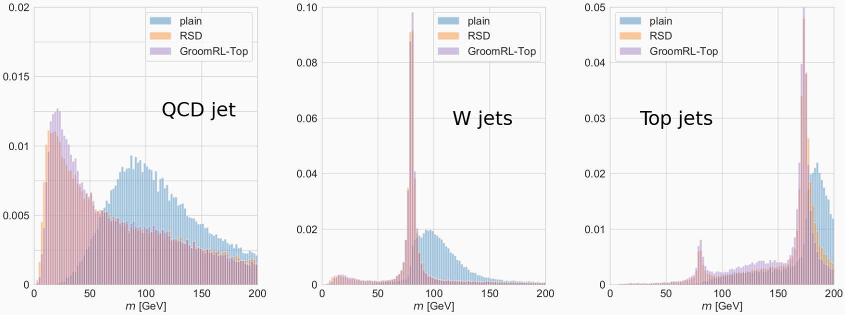


	$w_{\mathrm{max}} - w_{\mathrm{min}}$ [GeV]	w_{med} [GeV]
plain	44.65	104.64
`GroomRL-W`	10.70	80.09
`GroomRL-Top`	13.88	80.46
RSD	16.96	80.46

TABLE II: Size of the window containing 60% of the W mass spectrum, and median value on that interval.

Optimal GroomRL model for W jets

GroomRL-Top tested on QCD, W and Top jet data

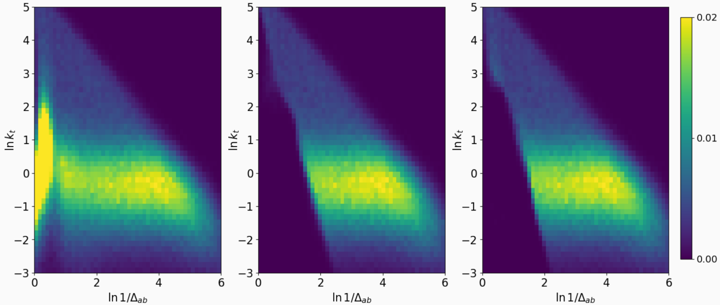


	$w_{\max} - w_{\min}$ [GeV]	w_{med} [GeV]
plain	44.65	104.64
GroomRL-W	10.70	80.09
GroomRL-Top	13.88	80.46
RSD	16.96	80.46

TABLE II: Size of the window containing 60% of the W mass spectrum, and median value on that interval.

Lund jet plane density

Lund jet plane before and after applying GroomRL



(a) Plain (b) GroomRL-W (c) GroomRL-Top

Inspecting $(\ln 1/\Delta_{ab},\, \ln k_t) \Rightarrow$ soft and wide-angle radiation removed.

Towards transfer learning in HEP

Towards transfer learning

Some ideas towards the "transfer learning" concept:

1. Generalize models trained on specific data to new datasets.

Towards transfer learning

Some ideas towards the "transfer learning" concept:

1. Generalize models trained on specific data to new datasets.
    - *e.g.* jet grooming

Towards transfer learning

Some ideas towards the "transfer learning" concept:

1. Generalize models trained on specific data to new datasets.
    - *e.g.* jet grooming
2. Build models specialized on the conversion between datasets.

Towards transfer learning

Some ideas towards the "transfer learning" concept:

1. Generalize models trained on specific data to new datasets.
   - *e.g.* jet grooming
2. Build models specialized on the conversion between datasets.
   - *e.g.* CycleGANs

Towards transfer learning

Some ideas towards the "transfer learning" concept:

1. Generalize models trained on specific data to new datasets.
   - *e.g.* jet grooming
2. Build models specialized on the conversion between datasets.
   - *e.g.* CycleGANs
3. Models that propagate higher-order correction to lower order.

Towards transfer learning

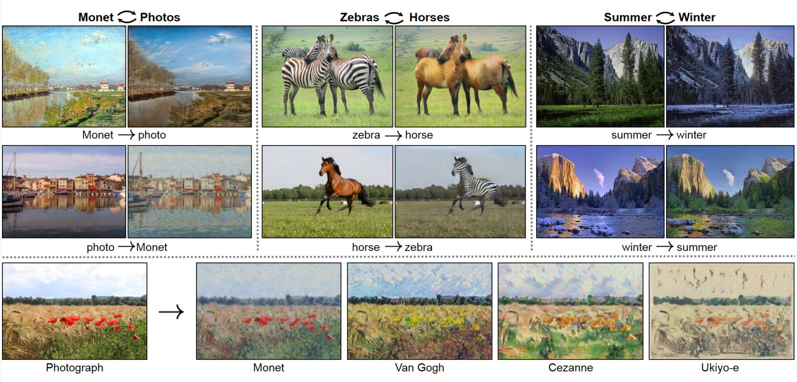
Some ideas towards the "transfer learning" concept:

1. Generalize models trained on specific data to new datasets.
   - *e.g.* jet grooming
2. Build models specialized on the conversion between datasets.
   - *e.g.* CycleGANs
3. Models that propagate higher-order correction to lower order.
   - *e.g.* reweighting

Cycle-consistent adversarial networks

[arXiv:1909.01359]

CycleGAN learns unpaired image-to-image mapping functions.

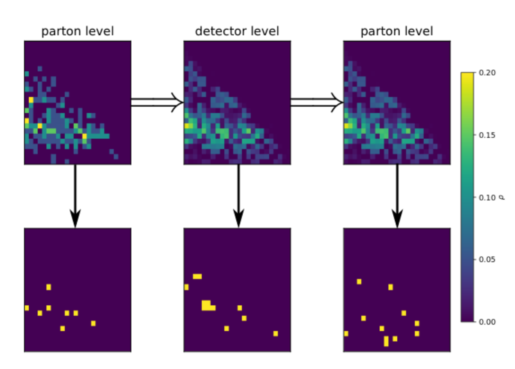


Reinterpreting events with CycleGANs

Use CycleGAN to transform between two different jet datasets, e.g.

- parton-level simulation $\leftrightarrow$ detector-level simulation
- W jet $\leftrightarrow$ QCD jet

Transformed events in good agreement with true sample.

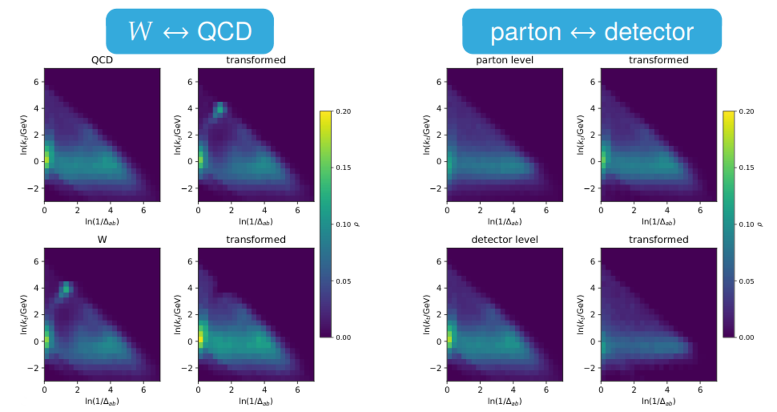


Reinterpreting events with CycleGANs

Use CycleGAN to transform between two different jet datasets, e.g.

- parton-level simulation $\leftrightarrow$ detector-level simulation
- W jet $\leftrightarrow$ QCD jet

Transformed events in good agreement with true sample.



MINLO t-channel single-top plus jet

[arXiv:1805.09855 - S.C., Frederix, Hamilton, Zanderighi '18]

Use neural nets to adjust unknown higher-order resummation terms.

Use NLO-matched single-top + jet (STJ) from the POWHEG-MINLO formalism:

$$d\sigma_{\mathcal{M}} = \Delta(y_{12}) \left[d\sigma_{\rm NLO}^{\rm STJ} - \Delta(y_{12})|_{\bar{\alpha}_S} d\sigma_{\rm LO}^{\rm STJ} \right]$$

MINLO t-channel single-top plus jet

[arXiv:1805.09855 - S.C., Frederix, Hamilton, Zanderighi '18]

Use neural nets to adjust unknown higher-order resummation terms.

Use NLO-matched single-top + jet (STJ) from the POWHEG-MINLO formalism:

$$d\sigma_{\mathcal{M}} = \Delta(y_{12}) \left[d\sigma^{\mathrm{STJ}}_{\mathrm{NLO}} - \Delta(y_{12})|_{\bar{\alpha}_S} d\sigma^{\mathrm{STJ}}_{\mathrm{LO}} \right]$$

Advantage: enhance fixed-order calculation with matched NLL Sudakov form factor.

MINLO t-channel single-top plus jet

[arXiv:1805.09855 - S.C., Frederix, Hamilton, Zanderighi '18]

Use neural nets to adjust unknown higher-order resummation terms.

Use NLO-matched single-top + jet (STJ) from the POWHEG-MINLO formalism:

$$d\sigma_{\mathcal{M}} = \Delta(y_{12}) \left[d\sigma_{\mathrm{NLO}}^{\mathrm{STJ}} - \Delta(y_{12})|_{\bar{\alpha}_S} d\sigma_{\mathrm{LO}}^{\mathrm{STJ}} \right]$$

Advantage: enhance fixed-order calculation with matched NLL Sudakov form factor.

Issue: this spoils the NLO accuracy of ST (single-top observables).

MINLO t-channel single-top plus jet

[arXiv:1805.09855 - S.C., Frederix, Hamilton, Zanderighi '18]

Use neural nets to adjust unknown higher-order resummation terms.

Use NLO-matched single-top + jet (STJ) from the POWHEG-MINLO formalism:

$$d\sigma_{\mathcal{M}} = \Delta(y_{12}) \left[d\sigma_{\rm NLO}^{\rm STJ} - \Delta(y_{12})|_{\bar{\alpha}_S} d\sigma_{\rm LO}^{\rm STJ} \right]$$

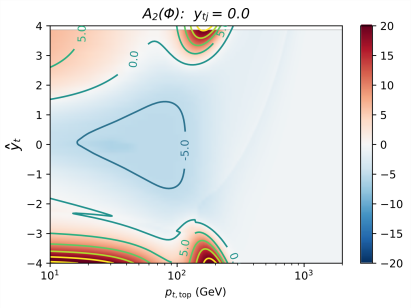
Advantage: enhance fixed-order calculation with matched NLL Sudakov form factor.

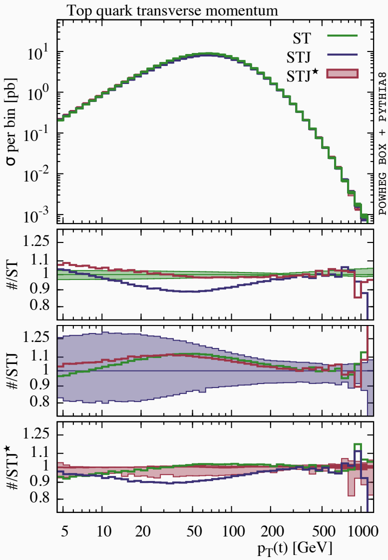
Issue: this spoils the NLO accuracy of ST (single-top observables).

Solution: fix at NNLL, fit A_2 with a Neural Network-based tuning of degrees of freedom, and test universality at 8 TeV.

$$\ln \delta\Delta(y_{12}) = -2 \int_{y_{12}}^{Q_{bt}^2} \frac{dq^2}{q^2} \bar{\alpha}_S^2 \mathcal{A}_2(\Phi) \ln \frac{Q_{bt}^2}{q^2}$$

MINLO t-channel single-top plus jet





Accelerating MC with ML tools

VEGAS integration algorithm and tensorflow

ML frameworks such as **TensorFlow** can help theoretical computations.

- Automatic parallelization and code optimization.
- Distributed computation across multiple hardware accelerators.

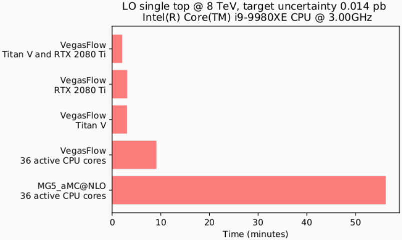
VEGAS integration algorithm and tensorflow

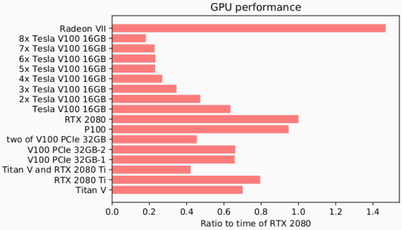
ML frameworks such as **TensorFlow** can help theoretical computations.

- Automatic parallelization and code optimization.
- Distributed computation across multiple hardware accelerators.

Example: VegasFlow [arXiv:1909.01359]

Monte Carlo integration using Vegas algorithm and TensorFlow code.





Thanks for your attention!