Jet grooming through reinforcement learning

based on PRD 100, 014014, arXiv:1903.09644

Stefano Carrazza and Frédéric Dreyer

QCD@LHC19, Buffalo, 16 July 2019.

Università degli Studi di Milano and INFN Sezione di Milano

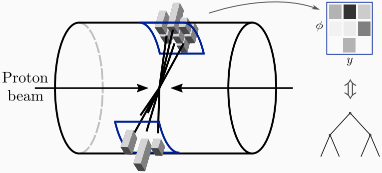
Acknowledgement: This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement number 740006.

Introduction

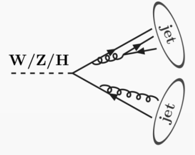
Boosted jets at the LHC

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

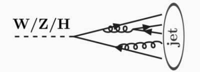
- particles such as W, Z, H, $t, \ldots$ are produced with $p_T^{\text{jet}} \gg m_{\text{jet}}$,
- hadronic collimated decays often reconstructed into single jets.



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



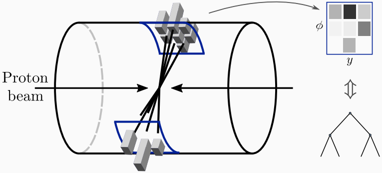
$p_T^{\text{jet}} \gg m_{\text{jet}}$



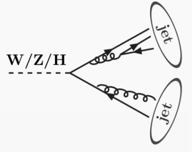
Boosted jets at the LHC

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

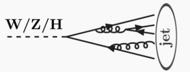
- particles such as W, Z, H, t, ... are produced with $p_T^{\text{jet}} \gg m_{\text{jet}}$,
- hadronic collimated decays often reconstructed into single jets.



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



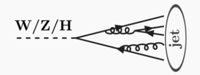
$p_T^{\text{jet}} \gg m_{\text{jet}}$



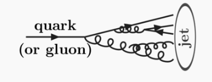
Problem: identify **hard structure** of a jet from **radiation patterns**.

(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

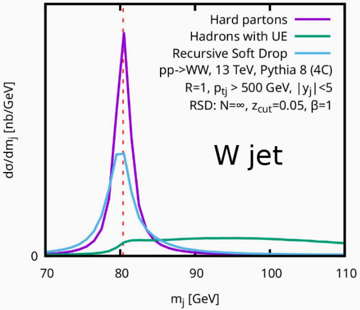
How to identify hard structure of a jet?

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

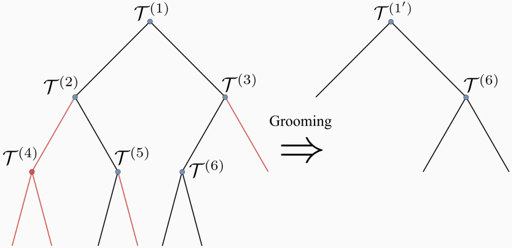
Grooming goal ⇒ remove unassociated soft wide-angle radiation.

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

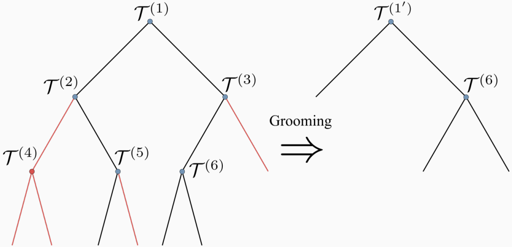


(Recursive) Soft Drop algorithm



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

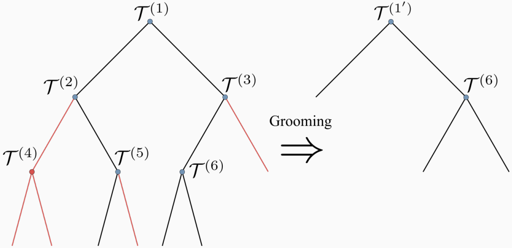
(Recursive) Soft Drop algorithm



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

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

(Recursive) Soft Drop algorithm



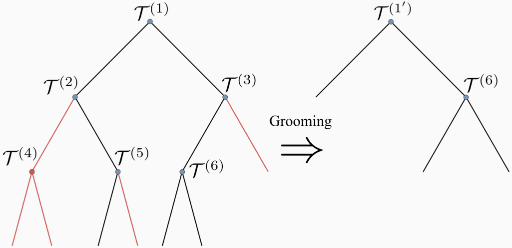
1. Cast jet as clustering tree with nodes $\mathcal{T}^{(i)}$ and children nodes a, b.
2. Define state of each node as $s_t = \{z, \Delta_{ab}\}$ where

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

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



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

❷ Define state of each node as $s_t = \{z, \Delta_{ab}\}$ where

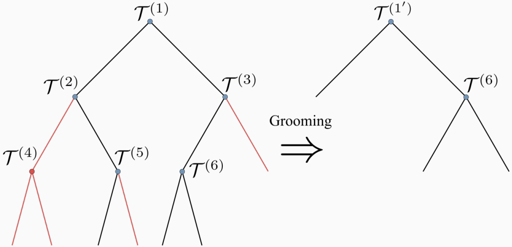
$$z = \frac{\min(p_{t,a}, p_{t,b})}{p_{t,a} + p_{t,b}}, \quad \Delta_{ab}^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2$$

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

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

❹ If $\pi_{\text{RSD}}(s_t) = 1 \rightarrow$ 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, $z_{\rm cut}$ and R_0 are free parameters):

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

4. If $\pi_{\rm RSD}(s_t) = 1 \to$ remove softer branch and update jet tree,
5. If $\pi_{\rm RSD}(s_t) = 0 \to$ 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:

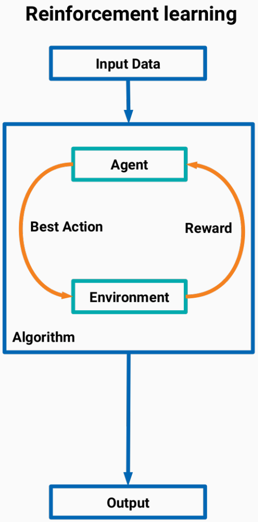
Generate jet events with Monte Carlo. Define a set of possible **states** in a five dimensional box:

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

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?

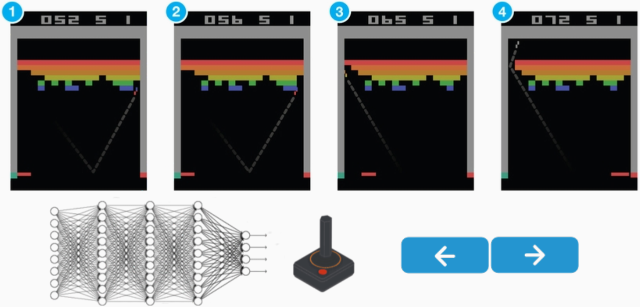
Deep Q-Network $\rightarrow$ off-policy and discrete action space.

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 $m_{\rm jet}$ 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)*

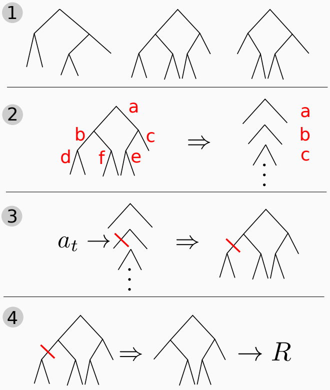
Environment

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:

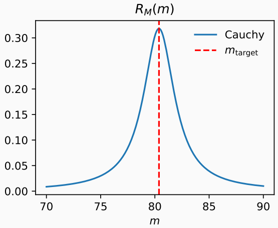
$$R(m, a_t, \Delta, z) = {\color{blue}R_M(m)} + \frac{1}{N_{\mathrm{SD}}} {\color{red}R_{\mathrm{SD}}(a_t, \Delta, z)}$$

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:

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



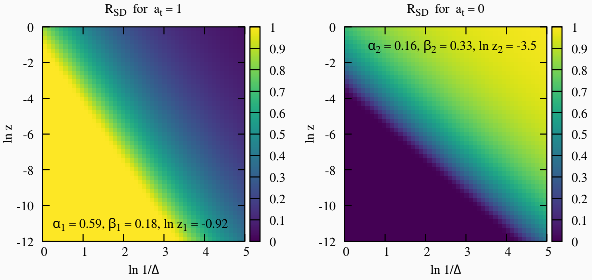
Reward function

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 ⇒ use multi-level training:

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

Adding a multi-level approach

What about background events?

Potential mass bias for background events ⇒ use multi-level training:

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

Adding a multi-level approach

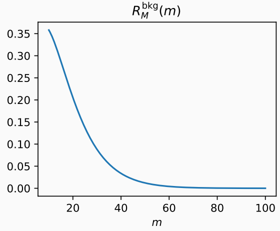
What about background events?

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 $\Rightarrow$ *(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.
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?

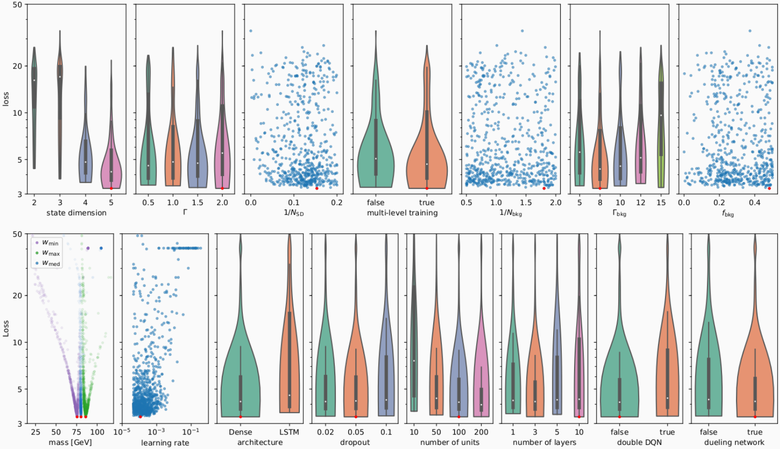
Use distributed asynchronous hyperparameter optimization ⇒ `hyperopt`.

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 $(w_{\min}, w_{\max})$ containing 60% of signal distribution,
   - the median w_{med} in that interval.
3. Define f_{bkg} the fraction of groomed background sample $(w_{\min}, w_{\max})$:

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

Hyperparameter tune

Validation loss for 2000 models



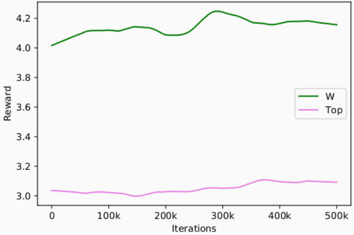
Results

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.

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

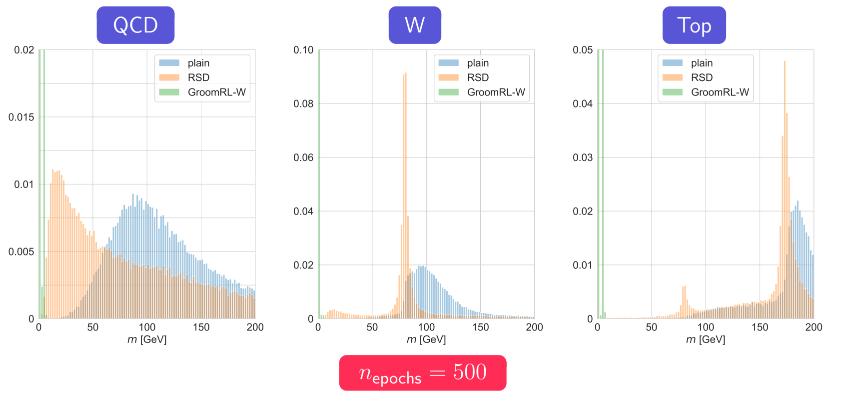


Parameters	Value
m_{target}	80.385 GeV or 173.2 GeV
s_t dimension	5
reward	Cauchy
Γ	2 GeV
$(\alpha_1, \beta_1, \ln z_1)$	$(0.59, 0.18, -0.92)$
$(\alpha_2, \beta_2, \ln z_2)$	$(0.65, 0.33, -3.53)$
$1/N_{\text{SD}}$	0.15
multi-level training	Yes
Γ_{bkg}	8 GeV
$1/N_{\text{bkg}}$	1.8 or 1.0
p_{bkg}	0.48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

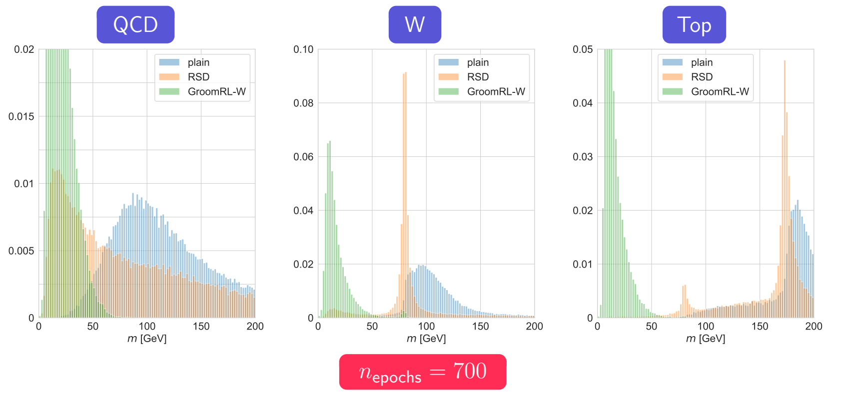
DRL training animation

GroomRL-W predictions vs n_{epochs}



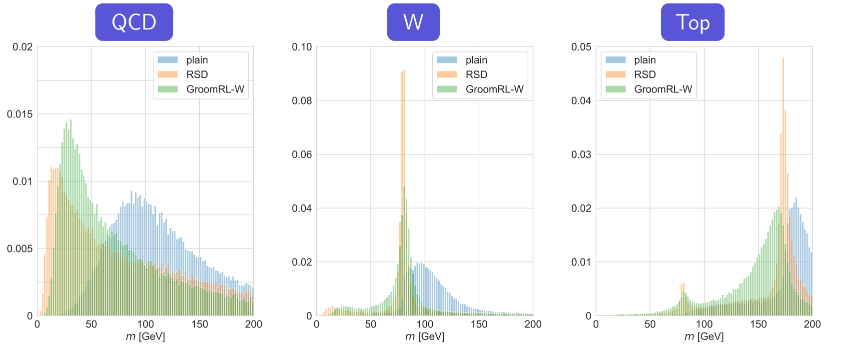
DRL training animation

GroomRL-W predictions vs n_{epochs}



DRL training animation

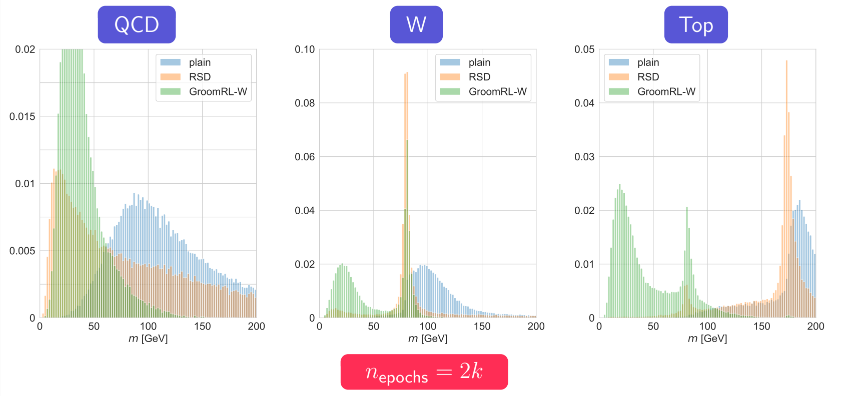
`GroomRL-W` predictions vs n_{epochs}



$n_{\text{epochs}} = 1k$

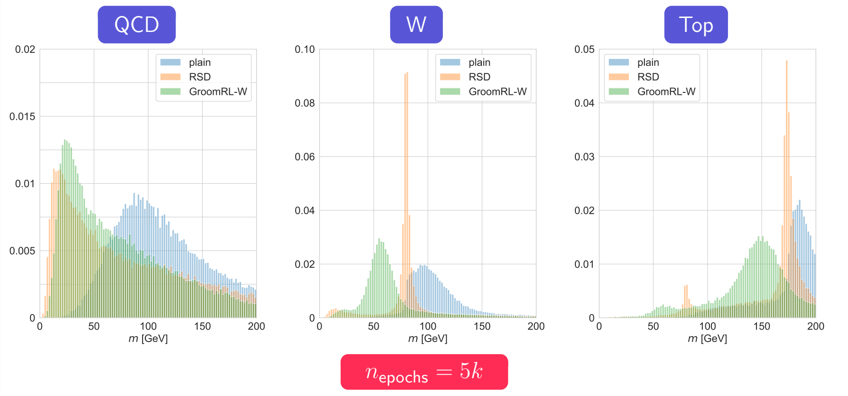
DRL training animation

GroomRL-W predictions vs n_{epochs}

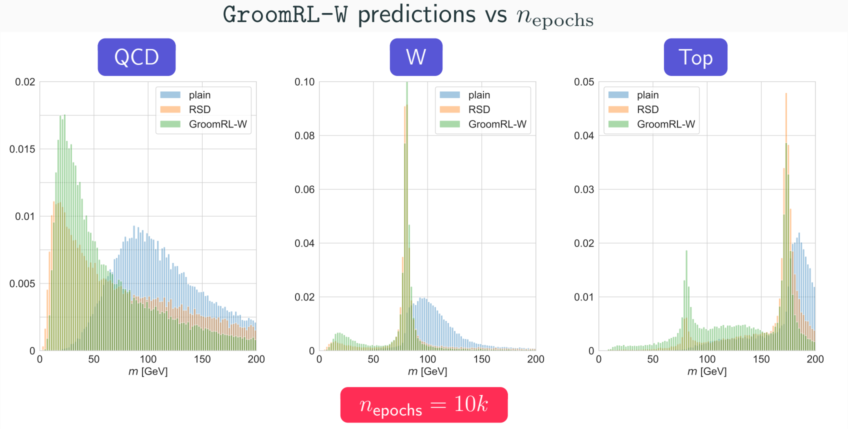


DRL training animation

GroomRL-W predictions vs n_{epochs}

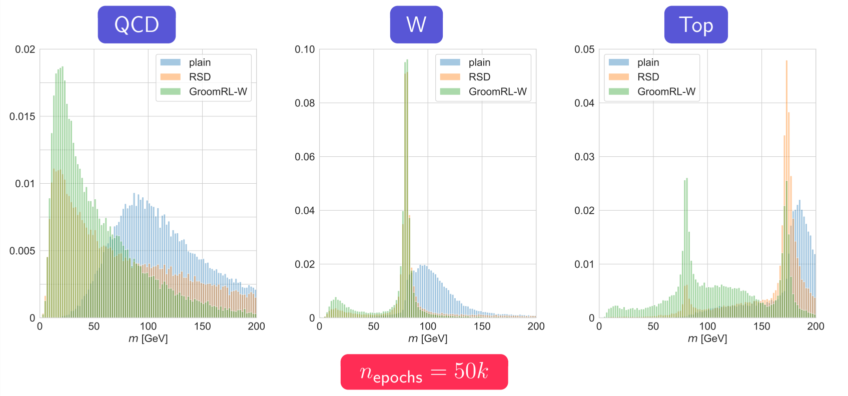


DRL training animation



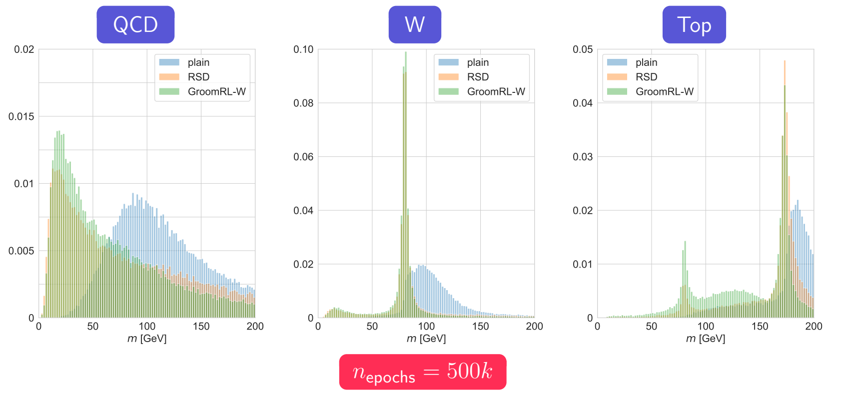
DRL training animation

GroomRL-W predictions vs n_{epochs}



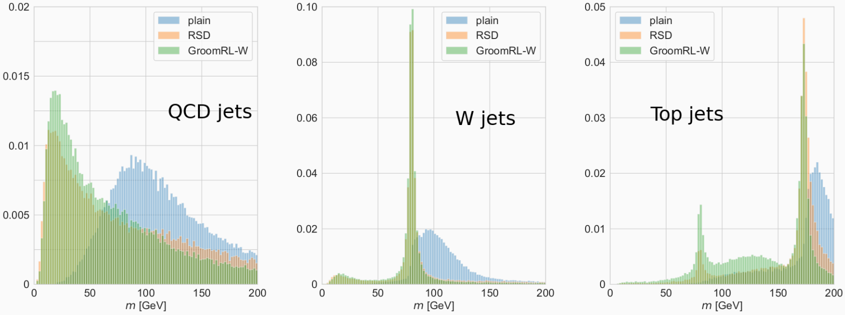
DRL training animation

`GroomRL-W` predictions vs n_{epochs}



Optimal GroomRL model for W jets

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

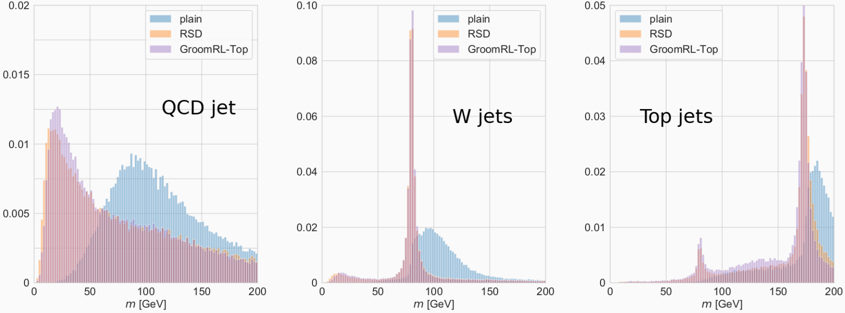


	$w_{\max} - w_{\min}$ [GeV]	w_{med} [GeV]
plain	44.65	104.64
GroomRL-W	10.70	80.09
GroomRL-Top	13.88	80.46
RSD	16.96	80.46

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

Optimal GroomRL model for W jets

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

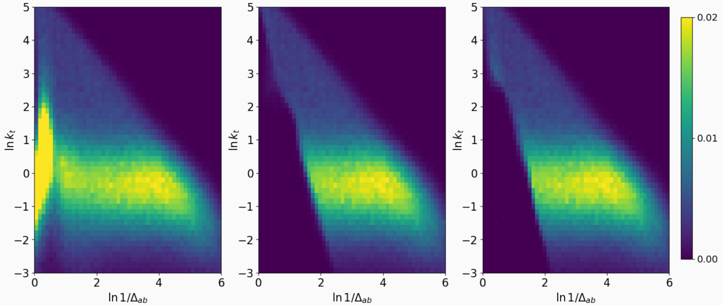


	$w_{\max} - w_{\min}$ [GeV]	w_{med} [GeV]
plain	44.65	104.64
GroomRL-W	10.70	80.09
GroomRL-Top	13.88	80.46
RSD	16.96	80.46

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.

Deliverables and conclusion

Deliverables and conclusions

Deliverables

- GroomRL complete python framework available at:
  https://github.com/JetsGame/GroomRL
  (contains pre-trained W and top jet DQN models)
- libGroomRL a C++ library for jet grooming models inference:
  https://github.com/JetsGame/libGroomRL
- Datasets for top, W and QCD jets at:
  https://github.com/JetsGame/data

Deliverables and conclusions

Deliverables

- GroomRL complete python framework available at:
  https://github.com/JetsGame/GroomRL
  (contains pre-trained W and top jet DQN models)
- libGroomRL a C++ library for jet grooming models inference:
  https://github.com/JetsGame/libGroomRL
- Datasets for top, W and QCD jets at:
  https://github.com/JetsGame/data

Conclusions

- Reinforcement learning can be applied to jet grooming successfully.
- Results are quantitatively similar to RSD with moderate improvement in mass resolution.
- Remarkable model generalization when changing underlying process without retraining.

Thank you!