Jet Grooming through Reinforcement Learning

Stefano Carrazza[1] **and Frédéric A. Dreyer**[2]

[1] TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Milan, Via Celoria 16, 20133, Milano, Italy
[2] Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Clarendon Laboratory, Parks Road, Oxford OX1 3PU, UK

E-mail: stefano.carrazza@cern.ch, frederic.dreyer@physics.ox.ac.uk

Abstract. We introduce a novel implementation of a reinforcement learning (RL) algorithm which is designed to find an optimal jet grooming strategy, a critical tool for collider experiments. The RL agent is trained with a reward function constructed to optimize the resulting jet properties, using both signal and background samples in a simultaneous multi-level training. We show that the grooming algorithm derived from the deep RL agent can match state-of-the-art techniques used at the Large Hadron Collider, resulting in improved mass resolution for boosted objects. Given a suitable reward function, the agent learns how to train a policy which optimally removes soft wide-angle radiation, allowing for a modular grooming technique that can be applied in a wide range of contexts. These results are accessible through the corresponding `GroomRL` framework.

1. Introduction

Jets are one of the most common objects appearing in proton-proton colliders such as the Large Hadron Collider (LHC) at CERN. They are defined as collimated bunches of high-energy particles, which emerge from the interactions of quarks and gluons, the fundamental constituents of the proton. In modern analyses, final-state particle momenta (i.e. the product of mass and velocity of the outgoing particles) are mapped to jet momenta using a sequential recombination algorithm with a single free parameter, the jet radius R, which defines up to which angle particles can get recombined into a given jet [1].

Due to the very high energies of its collisions, the LHC is routinely producing heavy particles with transverse momenta, the momentum component transverse to the beam axis, far greater than their rest mass. When these objects are sufficiently energetic (or boosted), they can often generate very collimated decay products, which are then reconstructed as a single fat jet, whose radiation patterns differ from standard quark or gluon jets. Since the advent of the LHC program, the study of the substructure of jets has matured into a remarkably active field of research that has become notably conducive to applications of recent Machine Learning techniques [2, 3, 4].

A particularly useful set of tools for experimental analyses are jet grooming algorithms, defined as a post-processing treatment of jets to remove unenergetic wide-angle radiation which is not associated with the underlying hard substructure. Grooming techniques play a crucial role in Standard Model measurements [5, 6] and in improving the boson- and top-tagging efficiencies at the LHC.

In this paper, we describe the `GroomRL` framework [7], which is used to train a grooming algorithm using reinforcement learning (RL). To this end, we decompose the problem of jet

Algorithm 1 Grooming

Input: policy π_g, binary tree node $\mathcal{T}^{(i)}$
$a_t = \pi_g(\mathcal{T}^{(i)} \to s_t)$
if $a_t == 1$ **then**
  $\mathcal{T}^{(j)} = \mathcal{T}^{(i)}$
  while $\mathcal{T}^{(j)} = (\mathcal{T}^{(j)} \to \text{parent})$ **do**
    $\mathcal{T}^{(j)} \to s_t = (\mathcal{T}^{(j)} \to s_t) - (\mathcal{T}^{(i)} \to b \to s_t)$
  end while
  $\mathcal{T}^{(i)} = (\mathcal{T}^{(i)} \to a)$
  Grooming(π_g, $\mathcal{T}^{(i)}$)
else
  Grooming(π_g, $\mathcal{T}^{(i)} \to a$)
  Grooming(π_g, $\mathcal{T}^{(i)} \to b$)
end if

grooming into successive steps for which a reward function can be designed taking into account the physical features that characterize such a system. We then use a modified implementation of a Deep Q-Network (DQN) agent [8, 9] and train a dense neural network (NN) to optimally remove radiation unassociated from the core of the jet. The trained model can then be applied on other data sets, showing improved resolution compared to state-of-the-art techniques as well as a strong resilience to non-perturbative effects. The framework and data used in this paper are available as open-source and published material in [10, 11].[1]

2. Jet representation

Let us start by introducing the representation we use for jets. We take the particle constituents of a jet, as defined by any modern algorithm, and recombine them using a Cambridge/Aachen (CA) sequential clustering algorithm [12]. The CA algorithm does a pairwise recombination, adding together the momenta of the two particles with the closest distance as defined by the measure $\Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$, where y_i is the rapidity, a measure of relativistic velocity along the beam axis, and ϕ_i is the azimuthal angle of particle i around the same axis. This clustering sequence is then used to recast the jet as a full binary tree, where each of the nodes contains information about the kinematic properties of the two parent particles. For each node i of the tree we define an object $\mathcal{T}^{(i)}$ containing the current observable state s_t, as well as a pointer to the two children nodes and one to the parent node. The children nodes a and b are ordered in transverse momentum such that $p_{t,a} > p_{t,b}$, and we label a the "harder" child and b the "softer" one. The set of possible states is defined by a five dimensional box, such that the state of the node is a tuple

$$s_t = \{z, \Delta_{ab}, \psi, m, k_t\} \,, \tag{1}$$

where $z = p_{t,b}/(p_{t,a} + p_{t,b})$ is the momentum fraction of the softer child b, $\psi = \tan^{-1}\left(\frac{y_b - y_a}{\phi_a - \phi_b}\right)$ is the azimuthal angle around the i axis, m is the mass, and $k_t = p_{t,b}\Delta_{ab}$ is the transverse momentum of b relative to a.

2.1. Grooming algorithm

A grooming algorithm acting on a jet tree can be defined by a simple recursive procedure which follows each of the branches and uses a policy $\pi_g(s_t)$ to decide based on the values of the current tuple s_t whether to remove the softer of the two branches. This is shown in Algorithm 1, where

[1] The code is available at `https://github.com/JetsGame/GroomRL`.

the minus sign is understood to mean the update of the kinematics of a node after removal of a soft branch. The grooming policy $\pi_g(s_t)$ returns an action $a_t \in \{0, 1\}$, with $a_t = 1$ corresponding to the removal of a branch, and $a_t = 0$ leaving the node unchanged. The state s_t is used to evaluate the current action-values $Q^*(s, a)$ for each possible action, which in turn are used to determine the best action at this step through a greedy policy.

It is easy to translate modern grooming algorithms in this language. For example, Recursive Soft Drop (RSD) [13] corresponds to a policy

$$\pi_{\text{RSD}}(s_t) = \begin{cases} 0 & \text{if} \quad z > z_{\text{cut}} \left(\frac{\Delta_{ab}}{R_0}\right)^{\beta}, \\ 1 & \text{else}, \end{cases} \tag{2}$$

where z_{cut}, β and R_0 are the parameters of the algorithm, and 1 corresponds as before to the action of removing the tree branch with smaller transverse momentum.

3. Setting up a grooming environment

In order to find an optimal grooming policy π_g, we introduce an environment and a reward function, formulating the problem in a way that can be solved using a RL algorithm.

We initialize a list of all trees used for the training, from which a tree is randomly selected at the beginning of each episode. Each step consists in taking the node with the largest Δ_{ab} value, following the CA declustering sequence, and taking an action on which of its branches to keep based on the state s_t of that node. Once a decision has been taken on the removal of the softer branch, and the parent nodes have been updated accordingly, the remaining children of the node are added to the list of nodes to consider in a following step of this episode. The reward function is then evaluated using the current state of the tree. The episode terminates once all nodes have been iterated over.

The framework described here deviates from usual RL implementations in that the range of possible states for any episode are fixed at the start. The transition probability between states $\mathcal{P}(s_{t+1}|s_t, a_t)$ therefore does not always depend very strongly on the action, although a grooming action can result in the removal of some of the future states and will therefore still have an effect on the distribution.

3.1. Finding optimal hyper-parameters

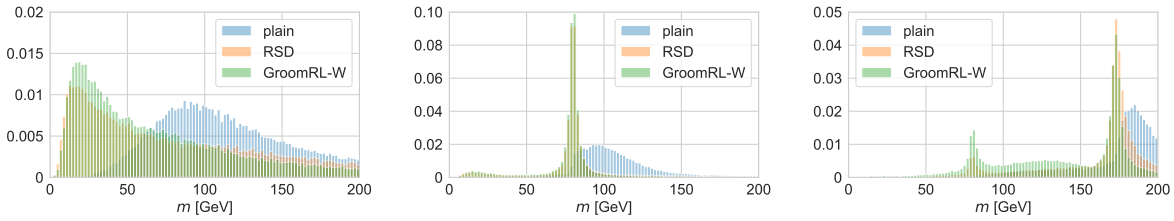
The optimal choice of hyper-parameters, both for the model architecture and for the grooming parameters, is determined using the distributed asynchronous hyper-parameter optimization library `hyperopt` [14].

The performance of an agent is evaluated by defining a loss function, which is evaluated on a distinct validation set consisting of 50k signal and background jets. For each sample, we evaluate the jet mass after grooming of each jet and derive the corresponding distribution. To calculate the loss function $\mathcal{L}$, we start by determining the smallest window $(w_{\text{min}}, w_{\text{max}})$ containing a fraction $f = 0.6$ of the final jet masses of the groomed signal distribution, defining w_{med} as the median value on that interval. The loss function is then defined as

$$\mathcal{L} = \frac{1}{5}|w_{\text{max}} - w_{\text{min}}| + |m_{\text{target}} - w_{\text{med}}| + 20 f_{\text{bkg}}, \tag{3}$$

where f_{bkg} is the fraction of the groomed background sample contained in the same interval, and m_{target} is a reference value for the signal.

We scan hyper-parameters using 1000 iterations and select the ones for which the loss $\mathcal{L}$ evaluated on the validation set is minimal. In practice we will do three different scans: to determine the best parameters of the reward function, to find an optimal grooming environment, and to determine the architecture of the DQN agent. The scan is performed by requiring



(a) QCD (b) W (c) top

Figure 1. Jet mass spectrum for (a) QCD jets, (b) W jets, (c) top jets. The `GroomRL-W` curve is obtained from training on W data.

`hyperopt` to use a uniform search space for continuous parameters, a log-uniform search space for the learning rate and a binary choice for all integer or boolean parameters. The optimization used in all the results presented in this work rely on the Tree-structured Parzen Estimator (TPE) algorithm [15].

3.2. Defining a reward function

One of the key ingredients for the optimization of the grooming policy is the reward function used at each step during the training. We consider a reward with two components: a first piece evaluated on the full tree, and another that considers only the kinematics of the current node.

The first component of the reward compares the mass of the current jet to a set target mass, typically the mass of the underlying boosted object. We implement this mass reward using a Cauchy distribution, which has two free parameters, the target mass m_{target} and a width Γ, so that

$$R_M(m) = \frac{\Gamma^2}{(|m - m_{\text{target}}|^2 + \Gamma^2)\pi}\,. \tag{4}$$

Separately, we calculate a reward on the current node which gives a positive reward for the removal of wide-angle soft radiation, as well as for leaving intact hard-collinear emissions. This provides a baseline behavior for the groomer. We label this reward component "Soft-Drop" due to its similarity with the Soft Drop condition [16], and implement it through exponential distributions

$$R_{\text{SD}}(a_t, \Delta, z) = a_t \min\left(1, e^{-\alpha_1 \ln(1/\Delta) + \beta_1 \ln(z_1/z)}\right) + (1 - a_t)\max\left(0, 1 - e^{-\alpha_2 \ln(1/\Delta) + \beta_2 \ln(z_2/z)}\right), \tag{5}$$

where $a_t = 0, 1$ is the action taken by the policy, and α_i, β_i, z_i are free parameters.

The total reward function is then given by

$$R(m, a_t, \Delta, z) = R_M(m) + \frac{1}{N_{\text{SD}}} R_{\text{SD}}(a_t, \Delta, z)\,. \tag{6}$$

Here N_{SD} is a normalization factor determining the weight given to the second component of the reward.

3.3. RL implementation and multi-level training

For the applications in this paper, we have implemented a DQN agent that contains a groomer module, which is defined by the underlying NN model and the test policy used by the agent. The groomer can be extracted after the model has been trained, using a greedy policy to select the best action based on the Q-values predicted by the NN. This allows for straightforward application of the resulting grooming strategy on new samples.

The training sample consists of 500k signal and background jets simulated using `Pythia` 8.223 [17]. We will construct two separate models by considering two signal samples, one with boosted W jets and one with boosted top jets, while the background always consists of QCD jets. We use the WW and $t\bar{t}$ processes, with hadronically decaying W and top, to create the signal samples, and the dijet process for the background. All samples used in this article can are available online [11]. The grooming environment is initialized by reading in the training data and creating an event array containing the corresponding jet trees.

To train the RL agent, we use a multi-level approach taking into account both signal and background samples. At the beginning of each episode, we select either a signal jet or a background jet, with probability $1 - p_{\text{bkg}}$. For signal jets, the reward function uses a reference mass set to the W-boson mass, $m_{\text{target}} = m_W$, or to the top mass, $m_{\text{target}} = m_t$, depending on the choice of sample. In the case of the background the mass reward function in equation (6) is changed to

$$R_M^{\text{bkg}}(m) = \frac{m}{\Gamma_{\text{bkg}}} \exp\Big(- \frac{m}{\Gamma_{\text{bkg}}} \Big) . \tag{7}$$

The width parameters Γ, Γ_{bkg} are also set to different values for signal and background reward functions, and are determined through a hyper-parameter scan.

We found that while this multi-level training only marginally improves the performance, it noticeably reduces the variability of the model.

4. Jet mass spectrum

Let us now apply the `GroomRL` models to new data samples. We consider three test sets of 50k elements each: one with QCD jets, one with W initiated jets and one with top jets. The size of the window containing 60% of the mass spectrum of the W sample, as well as the corresponding median value, are given in table 1 for each different grooming strategy. As a benchmark, we compare to the RSD algorithm, using parameters $z_{\text{cut}} = 0.05$, $\beta = 1$ and $R_0 = 1$. One can notice a sizeable reduction of the window size after grooming with the reinforcement learning based algorithms, while all groomers are able to reconstruct the peak location to a value very close to the W mass.

The distribution of the jet mass after grooming for each of these samples is shown in figure 1. Each curve gives the differential cross section $d\sigma/dm_j$ normalized by the total cross section. We show results for the grooming algorithm trained on a W sample, as well as for the ungroomed (or plain) jet mass and the jet mass after RSD grooming. As expected, one can observe that for the ungroomed case the resolution is very poor, with the QCD jets having large masses due to wide-angle radiation, while the W and top mass peaks are heavily distorted. In contrast, after applying RSD or `GroomRL`, the jet mass is reconstructed much more accurately. One interesting feature of `GroomRL` is that it is able to lower the jet mass for quark and gluon jets, further reducing the background contamination in windows close to a heavy particle mass.

In top jets, displayed in figure 1c, there are also noticeable enhancements after grooming with `GroomRL`, despite the fact that the training did not involve any top-related data. This demonstrates that the tools derived from our framework are robust and can be applied to data sets beyond their training range with good results.

5. Conclusions

We have shown a promising application of RL to the issue of jet grooming. Using a carefully designed reward function, we have constructed a groomer from a dense NN trained with a DQN agent.

This grooming algorithm was then applied to a range of data samples, showing excellent results for the mass resolution of boosted heavy particles. In particular, while the training of the

Table 1. Size of the window Δw containing 60% of the W mass spectrum, and median value on that interval.

	Plain	GroomRL-W	GroomRL-Top	RSD
Δw [GeV]	44.65	10.70	13.88	16.96
w_{med} [GeV]	104.64	80.09	80.46	80.46

NN is performed on samples consisting of W (or top) jets, the groomer yields noticeable gains in the top (or W) case as well, on data outside of the training range.

The improvements in resolution and background reduction compared to alternative state-of-the-art methods provide an encouraging demonstration of the relevance of machine learning for jet grooming. In particular, we showed that it is possible for a RL agent to extract the underlying physics of jet grooming and distill this knowledge into an efficient algorithm.

Due to its simplicity, the model we developed also retains most of the calculability of other existing methods such as Soft Drop. Accurate numerical computations of groomed jet observables are therefore achievable, allowing for the possibility of direct comparisons with data. Furthermore, given an appropriate sample, one could also attempt to train the grooming strategy on real data, bypassing some of the limitations due to the use of parton shower programs.

The `GroomRL` framework, available online [10], is generic and can easily be extended to higher-dimensional inputs, for example to consider multiple emissions per step or additional kinematic information. While the method presented in this article was applied to a specific problem in particle physics, we expect that with a suitable choice of reward function, this framework is in principle also applicable to a range of problems where a tree requires pruning.

Acknowledgments: We are grateful to Jia-Jie Zhu and Gavin Salam for comments on the manuscript and to Jesse Thaler for useful discussions. We also acknowledge the NVIDIA Corporation for the donation of a Titan Xp GPU used for this research. F.D. is supported by the Science and Technology Facilities Council (STFC) under grant ST/P000770/1. S.C. is supported by the European Research Council under the European Union's Horizon 2020 research and innovation Programme (grant agreement number 740006).

[1] Cacciari M, Salam G P and Soyez G 2008 *JHEP* **04** 063 (*Preprint* `0802.1189`)
[2] de Oliveira L, Kagan M, Mackey L, Nachman B and Schwartzman A 2016 *JHEP* **07** 069 (*Preprint* `1511.05190`)
[3] Louppe G, Cho K, Becot C and Cranmer K 2019 *JHEP* **01** 057 (*Preprint* `1702.00748`)
[4] Datta K and Larkoski A 2017 *JHEP* **06** 073 (*Preprint* `1704.08249`)
[5] Aaboud M *et al.* (ATLAS) 2018 *Phys. Rev. Lett.* **121** 092001 (*Preprint* `1711.08341`)
[6] Sirunyan A M *et al.* (CMS) 2018 *JHEP* **11** 113 (*Preprint* `1807.05974`)
[7] Carrazza S and Dreyer F A 2019 (*Preprint* `1903.09644`)
[8] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D and Riedmiller M A 2013 *CoRR* **abs/1312.5602** (*Preprint* `1312.5602`) URL `http://arxiv.org/abs/1312.5602`
[9] Mnih V, Kavukcuoglu K, Silver D, Rusu A A, Veness J, Bellemare M G, Graves A, Riedmiller M, Fidjeland A K, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S and Hassabis D 2015 *Nature* **518** 529–533 ISSN 00280836 URL `http://dx.doi.org/10.1038/nature14236`
[10] Carrazza S and Dreyer F A 2019 JetsGame/GroomRL v1.0.0 URL `https://doi.org/10.5281/zenodo.2602529`
[11] Carrazza S and Dreyer F A 2019 JetsGame/data v1.0.0 this repository is git-lfs. URL `https://doi.org/10.5281/zenodo.2602514`
[12] Dokshitzer Y L, Leder G D, Moretti S and Webber B R 1997 *JHEP* **08** 001 (*Preprint* `hep-ph/9707323`)
[13] Dreyer F A, Necib L, Soyez G and Thaler J 2018 *JHEP* **06** 093 (*Preprint* `1804.03657`)
[14] Bergstra J, Yamins D and Cox D D 2013 *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* ICML'13 (JMLR.org) pp I–115–I–123
[15] Bergstra J, Bardenet R, Bengio Y and Kégl B 2011 *Proceedings of the 24th International Conference on Neural Information Processing Systems* NIPS'11 (USA: Curran Associates Inc.) pp 2546–2554 ISBN 978-1-61839-599-3 URL `http://dl.acm.org/citation.cfm?id=2986459.2986743`

[16] Larkoski A J, Marzani S, Soyez G and Thaler J 2014 *JHEP* **05** 146 (*Preprint* 1402.2657)
[17] Sjöstrand T, Ask S, Christiansen J R, Corke R, Desai N, Ilten P, Mrenna S, Prestel S, Rasmussen C O and Skands P Z 2015 *Comput. Phys. Commun.* **191** 159–177 (*Preprint* 1410.3012)