QCD-Aware Recursive Neural Networks for Jet Physics

Gilles Louppe,[1] Kyunghyun Cho,[1] Cyril Becot,[1] and Kyle Cranmer[1]

[1]*New York University*

Recent progress in applying machine learning for jet physics has been built upon an analogy between calorimeters and images. In this work, we present a novel class of recursive neural networks built instead upon an analogy between QCD and natural languages. In the analogy, four-momenta are like words and the clustering history of sequential recombination jet algorithms is like the parsing of a sentence. Our approach works directly with the four-momenta of a variable-length set of particles, and the jet-based tree structure varies on an event-by-event basis. Our experiments highlight the flexibility of our method for building task-specific jet embeddings and show that recursive architectures are significantly more accurate and data efficient than previous image-based networks. We extend the analogy from individual jets (sentences) to full events (paragraphs), and show for the first time an event-level classifier operating on all the stable particles produced in an LHC event.

I. INTRODUCTION

By far the most common structures seen in collisions at the Large Hadron Collider (LHC) are collimated sprays of energetic hadrons referred to as ‘jets’. These jets are produced from the fragmentation and hadronization of quarks and gluons as described by quantum chromodynamics (QCD). Several goals for the LHC are centered around the treatment of jets, and there has been an enormous amount of effort from both the theoretical and experimental communities to develop techniques that are able to cope with the experimental realities while maintaining precise theoretical properties. In particular, the communities have converged on sequential recombination jet algorithms, methods to study jet substructure, and grooming techniques to provide robustness to pileup.

One compelling physics challenge is to search for highly boosted standard model particles decaying hadronically. For instance, if a hadronically decaying W boson is highly boosted, then its decay products will merge into a single fat jet with a characteristic substructure. Unfortunately, there is a large background from jets produced by more mundane QCD processes. For this reason, several jet ‘taggers’ and variables sensitive to jet substructure have been proposed. Initially, this work was dominated by techniques inspired by our intuition and knowledge of QCD; however, more recently there has been a wave of approaches that eschew this expert knowledge in favor of machine learning techniques. In this paper, we present a hybrid approach that leverages the structure of sequential recombination jet algorithms and deep neural networks.

Recent progress in applying machine learning techniques for jet physics has been built upon an analogy between calorimeters and images [1–8]. These methods take a variable-length set of 4-momenta and project them into a fixed grid of $\eta-\phi$ towers or ‘pixels’ to produce a ‘jet image’. The original jet classification problem, hence, reduces to an image classification problem, lending itself to deep convolutional networks and other machine learning algorithms. Despite their promising results, these models suffer from the fact that they have many free parameters and that they require large amounts of data for training. More importantly, the projection of jets into images also loses information, which impacts classification performance. The most obvious way to address this issue is to use a recurrent neural network to process a sequence of 4-momenta as they are. However, it is not clear how to order this sequence. While p_T ordering is common in many contexts [5], it does not capture important angular information critical for understanding the subtle structure of jets.

In this work, we propose instead a solution for jet classification based on an analogy between QCD and natural languages, as inspired by several works from natural language processing [9–14]. Much like a sentence is composed of words following a syntactic structure organized as a parse tree, a jet is also composed of 4-momenta following a structure dictated by QCD and organized via the clustering history of a sequential recombination jet algorithm. More specifically, our approach uses ‘recursive’ networks where the topology of the network is given by the clustering history of a sequential recombination jet algorithm, which varies on an event-by-event basis. This event-by-event adaptive structure can be contrasted with the ‘recurrent’ networks that operate purely on sequences (see e.g., [15]). The network is therefore given the 4-momenta without any loss of information, in a way that also captures substructures, as motivated by physical theory.

It is convenient to think of the recursive neural network as providing a ‘jet embedding’, which maps a set of 4-momenta into $\mathbb{R}^q$. This embedding has fixed length and can be fed into a subsequent network used for classification or regression. Thus the procedure can be used for jet tagging or estimating parameters that characterize the jet, such as the masses of resonances buried inside the jet. Importantly, the embedding and the subsequent network can be trained jointly so that the embedding is optimized for the task at hand.

Extending the natural language analogy paragraphs of text are sequence of sentences, just as event are sequence of jets. In particular, we propose to embed the full particle content of an event by feeding a sequence of jet-embeddings into a recurrent network. As before, this event-level embedding can be fed into a subsequent network used for classification or regression. To our knowledge, this represents the first machine learning model operating on all the detectable particles in an event.

The remainder of the paper is structured as follows. In Sec. II, we formalize the classification tasks at the jet-level and event-level. We describe the proposed recursive network architectures in Sec. III and detail the data samples and preprocessing used in our experiments in Sec. IV. Our results are summarized and discussed first in Sec. V for experiments on a jet-level classification problem, and then in Sec. VI for experiments on an event-level classification problem. In Sec. VII, we relate our work to close contributions from deep learning, natural language processing, and jet physics. Finally, we gather our conclusions and directions for further works in Sec. VIII.

II. PROBLEM STATEMENT

We describe a collision event $\mathbf{e} \in \mathcal{E}$ as being composed of a varying number of particles, indexed by i, and where each particle is represented by its 4-momentum vector $\mathbf{v}_i \in \mathbb{R}^4$, such that $\mathbf{e} = \{\mathbf{v}_i | i = 1, \ldots, N\}$.

The 4-momenta in each event can be clustered into jets with a sequential recombination jet algorithm that recursively combines (by simply adding their 4-momenta) the pair i, i' that minimize

$$d^{\alpha}_{ii'} = \min(p^{2\alpha}_{ti}, p^{2\alpha}_{ti'}) \frac{\Delta R^2_{ii'}}{R^2} \tag{1}$$

while $d^{\alpha}_{ii'}$ is less than $\min(p^{2\alpha}_{ti}, p^{2\alpha}_{ti'})$ [16, 17]. These sequential recombination algorithms have three hyper-parameters: R, $p_{t,\min}$, α, and jets with $p_t < p_{t,\min}$ are discarded. At that point, the jet algorithm has clustered $\mathbf{e}$ into M jets, each of which can be represented by a binary tree $\mathbf{t}_j \in \mathcal{T}$ indexed by $j = 1, \ldots, M$ with N_j leaves (corresponding to a subset of the $\mathbf{v}_i$). In the following, we will consider the specific cases where $\alpha = 1, 0, -1$, which respectively correspond to the k_t, Cambridge-Aachen and anti-k_t algorithms.

In addition to jet algorithms, we consider a 'random' baseline that corresponds to recombining particles at random to form random binary trees $\mathbf{t}_j$, along with 'asc-p_T' and 'desc-p_T' baselines, which correspond to degenerate binary trees formed from the sequences of particles sorted respectively in ascending and descending order of p_T.

For jet-level classification or regression, each jet $\mathbf{t}_j \in \mathcal{T}$ in the training data comes with labels or regression values $y_j \in \mathcal{Y}^{\text{jet}}$. In this framework, our goal is to build a predictive model $f^{\text{jet}} : \mathcal{T} \mapsto \mathcal{Y}^{\text{jet}}$ minimizing some loss function $\mathcal{L}^{\text{jet}}$. Similarly, for event-level classification or regression, we assume that each collision event $\mathbf{e}_l \in \mathcal{E}$ in the training data comes with labels or regression values $y_l \in \mathcal{Y}^{\text{event}}$, and our goal is to build a predictive model $f^{\text{event}} : \mathcal{E} \mapsto \mathcal{Y}^{\text{event}}$ minimizing some loss function $\mathcal{L}^{\text{event}}$.

III. RECURSIVE EMBEDDING

A. Individual jets

Let us first consider the case of an individual jet whose particles are topologically structured as a binary tree $\mathbf{t}_j$, e.g., based on a sequential recombination jet clustering algorithm or a simple sequential sorting in p_T. Let $k = 1, \ldots, 2N_j - 1$ indexes the node of the binary tree $\mathbf{t}_j$, and let the left and right children of node k be denoted by k_L and k_R respectively. Let also k_L always be the hardest child of k. By construction, we suppose that leaves k map to particles $i(k)$ while internal nodes correspond to recombinations. Using these notations, we recursively define the embedding $\mathbf{h}^{\text{jet}}_k \in \mathbb{R}^q$ of node k in $\mathbf{t}_j$ as

$$\mathbf{h}^{\text{jet}}_k = \begin{cases} \mathbf{u}_k & \text{if } k \text{ is a leaf} \\ \sigma\left(W_h \begin{bmatrix} \mathbf{h}^{\text{jet}}_{k_L} \\ \mathbf{h}^{\text{jet}}_{k_R} \\ \mathbf{u}_k \end{bmatrix} + b_h \right) & \text{otherwise} \end{cases} \tag{2}$$

$$\mathbf{u}_k = \sigma\left(W_u g(\mathbf{o}_k) + b_u\right) \tag{3}$$

$$\mathbf{o}_k = \begin{cases} \mathbf{v}_{i(k)} & \text{if } k \text{ is a leaf} \\ \mathbf{o}_{k_L} + \mathbf{o}_{k_R} & \text{otherwise} \end{cases} \tag{4}$$

where $W_h \in \mathbb{R}^{q \times 3q}$, $b_h \in \mathbb{R}^q$, $W_u \in \mathbb{R}^{q \times 4}$ and $b_u \in \mathbb{R}^q$ form together the shared parameters to be learned, q is the size of the embedding, σ is the ReLU activation function [18], and g is a function extracting the kinematic features p, η, θ, ϕ, E, and p_T from the 4-momentum $\mathbf{o}_k$.

When applying Eqn. 2 recursively from the root node $k = 1$ down to the outer nodes of the binary tree $\mathbf{t}_j$, the resulting embedding, denoted $\mathbf{h}^{\text{jet}}_1(\mathbf{t}_j)$, effectively summarizes the information contained in the particles forming the jet into a single vector. In particular, this recursive neural network (RNN) embeds a binary tree of varying shape and size into a vector of fixed size. As a result, the embedding $\mathbf{h}^{\text{jet}}_1(\mathbf{t}_j)$ can now be chained to a subsequent classifier or regressor to solve our target supervised learning problem, as illustrated in Figure 1. All parameters (i.e., of the recursive jet embedding and of the classifier) are learned jointly using backpropagation through structure [9] to minimize the loss $\mathcal{L}^{\text{jet}}$, hence tailoring the embedding to the specific requirements of the task. Further implementation details, including an efficient batched computation over distinct binary trees, can be found in Appendix C.

In addition to the recursive activation of Eqn. 2, we also consider and study its extended version equipped with reset and update gates (see details in Appendix A). This gated architecture allows the network to preferentially pass information along the left-child, right-child, or their combination.

While we have not performed experiments, we point out that there is an analogous style of architectures based on jet algorithms with 2 → 3 recombinations [17, 19, 20].

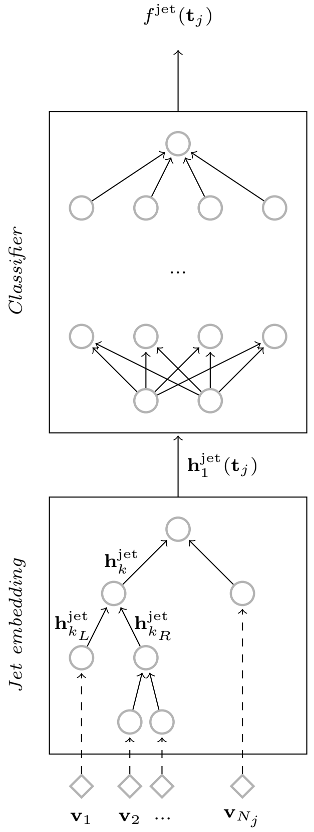
$f^{\mathrm{jet}}(\mathbf{t}_j)$
Classifier
...
$\mathbf{h}_1^{\mathrm{jet}}(\mathbf{t}_j)$
Jet embedding
$\mathbf{h}_k^{\mathrm{jet}}$
$\mathbf{h}_{k_L}^{\mathrm{jet}}$
$\mathbf{h}_{k_R}^{\mathrm{jet}}$
$\mathbf{v}_1$
$\mathbf{v}_2$
...
$\mathbf{v}_{N_j}$

FIG. 1. QCD-motivated recursive jet embedding for classification. For each individual jet, the embedding $\mathbf{h}_1^{\mathrm{jet}}(\mathbf{t}_j)$ is computed recursively from the root node down to the outer nodes of the binary tree $\mathbf{t}_j$. The resulting embedding is chained to a subsequent classifier, as illustrated in the top part of the figure. The topology of the network in the bottom part is distinct for each jet and is determined by a sequential recombination jet algorithm (e.g., k_t clustering).

B. Full events

We now embed entire events $\mathbf{e}$ of variable size by feeding the embeddings of their individual jets to an event-level sequence-based recurrent neural network.

As an illustrative example, we consider here a gated recurrent unit [21] (GRU) operating on the p_T ordered sequence of pairs $(\mathbf{v}(\mathbf{t}_j), \mathbf{h}_1^{\mathrm{jet}}(\mathbf{t}_j))$, for $j = 1, \ldots, M$, where $\mathbf{v}(\mathbf{t}_j)$ is the unprocessed 4-momentum of the jet $\mathbf{t}_j$ and $\mathbf{h}_1^{\mathrm{jet}}(\mathbf{t}_j)$ is its embedding. The final output $\mathbf{h}_M^{\mathrm{event}}(\mathbf{e})$ (see Appendix B for details) of the GRU is chained to a subsequent classifier to solve an event-level classification task. Again, all parameters (i.e., of the inner jet embedding function, of the GRU, and of the classifier) are learned jointly using backpropagation through structure [9] to minimize the loss $\mathcal{L}^{\mathrm{event}}$. Figure 2 provides a schematic of the full classification model. In summary, combining two levels of recurrence provides a QCD-motivated event-level embedding that effectively operates at the hadron-level for all the particles in the event.

In addition and for the purpose of comparison, we also consider the simpler baselines where i) only the 4-momenta $\mathbf{v}(\mathbf{t}_j)$ of the jets are given as input to the GRU, without augmentation with their embeddings, and ii) the 4-momenta $\mathbf{v}_i$ of the constituents of the event are all directly given as input to the GRU, without grouping them into jets or providing the jet embeddings.

IV. DATA, PREPROCESSING AND EXPERIMENTAL SETUP

In order to focus attention on the impact of the network architectures and the projection of input 4-momenta into images, we consider the same boosted W tagging example as used in Refs. [1, 2, 4, 6]. The signal $(y = 1)$ corresponds to a hadronically decaying W boson with $200 < p_T < 500$ GeV, while the background $(y = 0)$ corresponds to a QCD jet with the same range of p_T.

We are grateful to the authors of Ref. [6] for sharing the data used in their studies. We obtained both the full-event records from their PYTHIA benchmark samples, including both the particle-level data and the towers from the DELPHES detector simulation. In addition, we obtained the fully processed jet images of 25×25 pixels, which include the initial $R = 1$ anti-k_t jet clustering and subsequent trimming, translation, pixelisation, rotation, reflection, cropping, and normalization preprocessing stages detailed in Ref. [2, 6].

Our training data was collected by sampling from the original data a total of 100,000 signal and background jets with equal prior. The testing data was assembled similarly by sampling 100,000 signal and background jets, without overlap with the training data. For direct comparison with Ref. [6], performance is evaluated at test time within the restricted window of $250 < p_T < 300$ and $50 \leq m \leq 110$, where the signal and background jets are re-weighted to produce flat p_T distributions. Results are reported in terms of the area under the ROC curve (ROC AUC) and of background rejection (i.e., 1/FPR) at 50% signal efficiency ($R_{\epsilon=50\%}$). Average scores reported include uncertainty estimates that come from training 30 models with distinct initial random seeds. About 2% of the models had technical problems during training (e.g., due to numerical errors), so we applied a simple algorithm to ensure robustness: we discarded models whose $R_{\epsilon=50\%}$ was outside of 3 standard deviations of the mean, where the mean and standard deviation were estimated excluding the five best and worst performing models.

For our jet-level experiments we consider as input to the classifiers the 4-momenta $\mathbf{v}_i$ from both the particle-level data and the DELPHES towers. We also compare the performance with and without the projection of those 4-momenta into images. While the image data already included the full pre-processing steps, when considering particle-level and tower inputs we performed the initial $R = 1$ anti-k_t jet clustering to identify the constituents of the highest p_T jet $\mathbf{t}_1$ of each event, and then performed

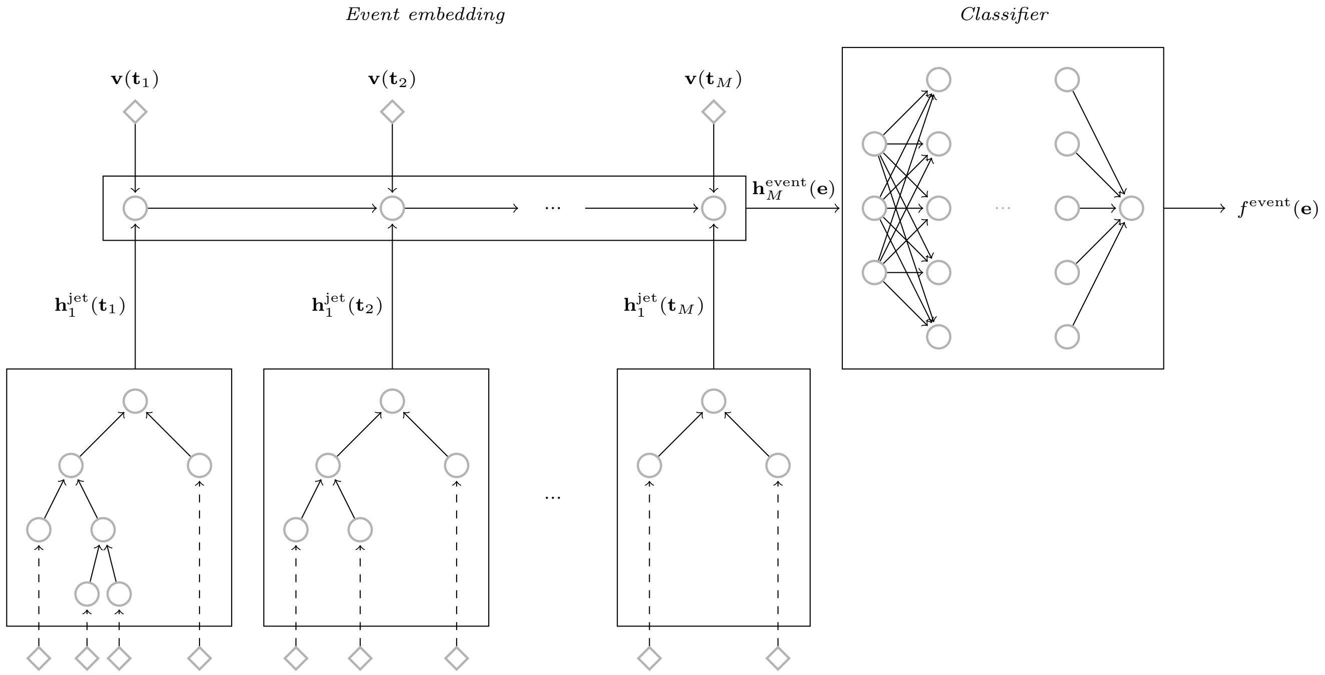
Event embedding
Classifier
$\mathbf{v}(\mathbf{t}_1)$
$\mathbf{v}(\mathbf{t}_2)$
$\mathbf{v}(\mathbf{t}_M)$
$\mathbf{h}_M^{\text{event}}(\mathbf{e})$
$f^{\text{event}}(\mathbf{e})$
$\mathbf{h}_1^{\text{jet}}(\mathbf{t}_1)$
$\mathbf{h}_1^{\text{jet}}(\mathbf{t}_2)$
$\mathbf{h}_1^{\text{jet}}(\mathbf{t}_M)$
...

FIG. 2. QCD-motivated event embedding for classification. The embedding of an event is computed by feeding the sequence of pairs $(\mathbf{v}(\mathbf{t}_j), \mathbf{h}_1^{\text{jet}}(\mathbf{t}_j))$ over the jets it is made of, where $\mathbf{v}(\mathbf{t}_j)$ is the unprocessed 4-momentum of the jet $\mathbf{t}_j$ and $\mathbf{h}_1^{\text{jet}}(\mathbf{t}_j)$ is its embedding. The resulting event-level embedding $\mathbf{h}_M^{\text{event}}(\mathbf{e})$ is chained to a subsequent classifier, as illustrated in the right part of the figure.

the subsequent translation, rotation, and reflection preprocessing steps (omitting cropping and normalization). When processing the image data, we inverted the normalization that enforced the sum of the squares of the pixel intensities be equal to one.[1]

For our event-level experiments we were not able to use the data from Ref. [6] because the signal sample corresponded to $pp \to W(\to J)Z(\to \nu\bar{\nu})$ and the background to $pp \to jj$. Thus the signal was characterized by one high-p_T jet and large missing energy from $Z(\to \nu\bar{\nu})$ which is trivially separated from the dijet background. For this reason, we generated our own PYTHIA and DELPHES samples of $pp \to W' \to W(\to J)Z(\to J)$ and QCD background such that both the signal and background have two high-p_T jets. We use $m_{W'} = 700$ GeV and restrict $\hat{p}_t$ of the $2 \to 2$ scattering process to $300 < \hat{p}_t < 350$ GeV. Our focus is to demonstrate the scalability of our method to all the particles or towers in an event, and not to provide a precise statement about physics reach for this signal process. In this case each event $\mathbf{e}$ was clustered by the same anti-k_t algorithm with $R = 1$, and then the constituents of each jet were treated as in Sec. III A (i.e., reclustered using k_t or a sequential ordering in p_T to provide the network topology for a non-gated embedding). Additionally, the constituents of each jet were pre-processed with translation, rotation, and reflection as in the individual jet case. Training was carried out on a dataset of 100,000 signal and background events with equal prior. Performance was evaluated on an independent test set of 100,000 other events, as measured by the ROC AUC and $R_{\epsilon=80\%}$ of the model predictions. Again, average scores are given with uncertainty estimates that come from training 30 models with distinct initial random seeds.

In both jet-level and event-level experiments, the dimension of the embeddings q was set to 40. Training was conducted using Adam [22] as an optimizer for 25 epochs, with a batch size of 64 and a learning rate of 0.0005 decayed by a factor of 0.9 after every epoch. These parameters were found to perform best on average, as determined through an optimization of the hyper-parameters. Performance was monitored during training on a validation set of 5000 samples to allow for early stopping and prevent from overfitting.

V. EXPERIMENTS WITH JET-LEVEL CLASSIFICATION

A. Performance studies

We carried out performance studies where we varied the following factors: the projection of the 4-momenta into an image, the source of those 4-momenta, the topol-

[1] In Ref. [2], the jet images did not include the DELPHES detector simulation, they were comparable to our *particle* scenario with the additional discretization into pixels.

TABLE I. Summary of jet classification performance for several approaches applied either to particle-level inputs or towers from a DELPHES simulation.

Input	Architecture	ROC AUC	$R_{\epsilon=50\%}$
Projected into images			
towers	MaxOut	**0.8418**	–
towers	k_t	0.8321 ± 0.0025	$\mathbf{12.7 \pm 0.4}$
towers	k_t (gated)	0.8277 ± 0.0028	12.4 ± 0.3
Without image preprocessing			
towers	τ_{21}	0.7644	6.79
towers	mass + τ_{21}	0.8212	11.31
towers	k_t	0.8807 ± 0.0010	24.1 ± 0.6
towers	C/A	0.8831 ± 0.0010	24.2 ± 0.7
towers	anti-k_t	0.8737 ± 0.0017	22.3 ± 0.8
towers	asc-p_T	0.8835 ± 0.0009	$\mathbf{26.2 \pm 0.7}$
towers	desc-p_T	$\mathbf{0.8838 \pm 0.0010}$	25.1 ± 0.6
towers	random	0.8704 ± 0.0011	20.4 ± 0.3
particles	k_t	0.9185 ± 0.0006	68.3 ± 1.8
particles	C/A	$\mathbf{0.9192 \pm 0.0008}$	68.3 ± 3.6
particles	anti-k_t	0.9096 ± 0.0013	51.7 ± 3.5
particles	asc-p_T	0.9130 ± 0.0031	52.5 ± 7.3
particles	desc-p_T	0.9189 ± 0.0009	$\mathbf{70.4 \pm 3.6}$
particles	random	0.9121 ± 0.0008	51.1 ± 2.0
With gating (see Appendix A)			
towers	k_t	0.8822 ± 0.0006	25.4 ± 0.4
towers	C/A	0.8861 ± 0.0014	26.2 ± 0.8
towers	anti-k_t	0.8804 ± 0.0010	24.4 ± 0.4
towers	asc-p_T	0.8849 ± 0.0012	27.2 ± 0.8
towers	desc-p_T	$\mathbf{0.8864 \pm 0.0007}$	$\mathbf{27.5 \pm 0.6}$
towers	random	0.8751 ± 0.0029	22.8 ± 1.2
particles	k_t	0.9195 ± 0.0009	74.3 ± 2.4
particles	C/A	$\mathbf{0.9222 \pm 0.0007}$	81.8 ± 3.1
particles	anti-k_t	0.9156 ± 0.0012	68.3 ± 3.2
particles	asc-p_T	0.9137 ± 0.0046	54.8 ± 11.7
particles	desc-p_T	0.9212 ± 0.0005	$\mathbf{83.3 \pm 3.1}$
particles	random	0.9106 ± 0.0035	50.7 ± 6.7

ogy of the RNN, and the presence or absence of gating.

a. Impact of image projection The first factor we studied was whether or not to project the 4-momenta into an image as in Refs. [2, 6]. The architectures used in previous studies required a fixed input (image) representation, and cannot be applied to the variable length set of input 4-momenta. Conversely, we can apply the RNN architecture to the discretized image 4-momenta. Table I shows that the RNN architecture based on a k_t topology performs almost as well as the MaxOut architecture in Ref. [6] when applied to the image pre-processed 4-momenta coming from DELPHES towers. Importantly the RNN architecture is much more data efficient. While the MaxOut architecture in Ref. [6] has 975,693 parameters and was trained with 6M examples, the non-gated RNN architecture has 8,481 parameters and was trained with 100,000 examples only.

Next, we compare the RNN classifier based on a k_t topology on tower 4-momenta with and without image preprocessing. Table I and Fig. 3 show significant gains in not using jet images, improving ROC AUC from 0.8321 to 0.8807 (resp., $R_{\epsilon=50\%}$ from 12.7 to 24.1) in the case of k_t topologies. In addition, this result outperforms the MaxOut architecture operating on images by a significant margin. This suggests that the projection into an image loses information and impacts classification performance. We suspect the loss of information to be due to some of the construction steps of jet images (i.e., pixelisation, rotation, zooming, cropping and normalization). In particular, all are applied at the image-level instead of being performed directly on the 4-momenta, which might induce artefacts due to the lower resolution, particle superposition and aliasing. By contrast, the RNN is able to work directly with the 4-momenta of a variable-length set of particles, without any loss of information. For completeness, we also compare to the performance of a classifier based purely on the single n-subjettiness feature $\tau_{21} := \tau_2/\tau_1$ and a classifier based on two features (the trimmed mass and τ_{21}) [23]. In agreement with previous results based on deep learning [2, 6], we see that our RNN classifier clearly outperforms this variable.

b. Measurements of the 4-momenta The second factor we varied was the source of the 4-momenta. The *towers* scenario, corresponds to the case where the 4-momenta come from the calorimeter simulation in DELPHES. While the calorimeter simulation is simplistic, the granularity of the towers is quite large (10° in ϕ) and it does not take into account that tracking detectors can provide very accurate momenta measurements for charged particles that can be combined with calorimetry as in the particle flow approach. Thus, we also consider the *particles* scenario, which corresponds to an idealized case where the 4-momenta come from perfectly measured stable hadrons from PYTHIA. Table I and Fig. 3 show that further gains could be made with more accurate measurements of the 4-momenta, improving e.g. ROC AUC from 0.8807 to 0.9185 (resp., $R_{\epsilon=50\%}$ from 24.1 to 68.3) in the case of k_t topologies. We also considered a case where the 4-momentum came from the DELPHES particle flow simulation and the data associated with each particle was augmented with a particle-flow identifier distinguishing ± charged hadrons, photons, and neutral hadrons. This is similar in motivation to Ref. [7], but we did not observe any significant gains in classification performance with respect to the *towers* scenario.

c. Topology of the binary trees The third factor we studied was the topology of the binary tree $\mathbf{t}_j$ described in Sections II and III A that dictates the recursive structure of the RNN. We considered binary trees based on the anti-k_t, Cambridge-Aachen (C/A), and k_t sequential recombination jet algorithms, along with random, asc-p_T and desc-p_T binary trees. Table I and Fig. 4 show the performance of the RNN classifier based on these various topologies. Interestingly, the topology is significant.

For instance, k_t and C/A significantly outperform the anti-k_t topology on both tower and particle inputs. This is consistent with intuition from previous jet substructure studies where jets are typically reclustered with the k_t algorithm. The fact that the topology is important is

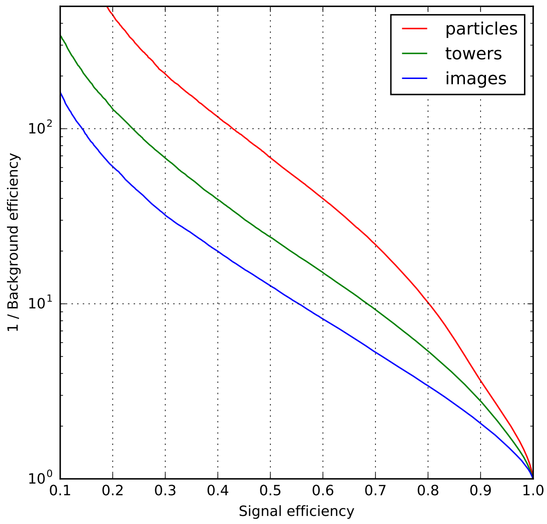

FIG. 3. Jet classification performance for various input representations of the RNN classifier, using k_t topologies for the embedding. The plot shows that there is significant improvement from removing the image processing step and that significant gains can be made with more accurate measurements of the 4-momenta.

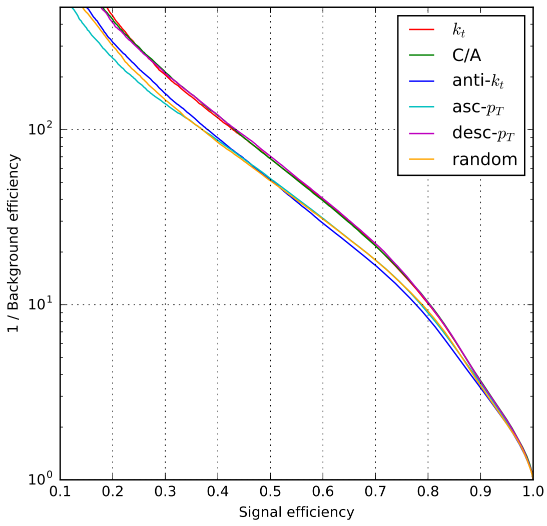

FIG. 4. Jet classification performance of the RNN classifier based on various network topologies for the embedding (*particles* scenario). This plot shows that topology is significant, as supported by the fact that results for k_t, C/A and desc-p_T topologies improve over results for anti-k_t, asc-p_T and random binary trees. Best results are achieved for C/A and desc-p_T topologies, depending on the metric considered.

further supported by the poor performance of the random binary tree topology. We expected however that a simple sequence (represented as a degenerate binary tree) based on ascending and descending p_T ordering would not perform particularly well, particularly since the topology does not use any angular information. Surprisingly, the simple descending p_T ordering slightly outperforms the RNNs based on k_t and C/A topologies. The descending p_T network has the highest p_T 4-momenta near the root of the tree, which we expect to be the most important. We suspect this is the reason that the descending p_T outperforms the ascending p_T ordering on particles, but this is not supported by the performance on towers. A similar observation was already made in the context of natural languages [24–26], where tree-based models have at best only slightly outperformed simpler sequence-based networks. While recursive networks appear as a principled choice, it is conjectured that recurrent networks may in fact be able to discover and implicitly use recursive compositional structure by themselves, without supervision.

d. Gating The last factor that we varied was whether or not to incorporate gating in the RNN. Adding gating increases the number of parameters to 48,761, but this is still about 20 times smaller than the number of parameters in the MaxOut architectures used in previous jet image studies. Table I shows the performance of the various RNN topologies with gating. While results improve significantly with gating, most notably in terms of $R_{\epsilon=50\%}$, the trends in terms of topologies remain unchanged.

e. Other variants Finally, we also considered a number of other variants. For example, we jointly trained a classifier with the concatenated embeddings obtained over k_t and anti-k_t topologies, but saw no significant performance gain. We also tested the performance of recursive activations transferred across topologies. For instance, we used the recursive activation learned with a k_t topology when applied to an anti-k_t topology and observed a significant loss in performance. We also considered particle and tower level inputs with an additional trimming preprocessing step, which was used for the jet image studies, but we saw a significant loss in performance. While the trimming degraded classification performance, we did not evaluate the robustness to pileup that motivates trimming and other jet grooming procedures.

B. Infrared and Collinear Safety Studies

In proposing variables to characterize substructure, physicists have been equally concerned with classification performance and the ability to ensure various theoretical properties of those variables. In particular, initial work on jet algorithms focused on the Infrared-Collinear (IRC) safe conditions:

- *Infrared safety.* The model is robust to augmenting $\mathbf{e}$ with additional particles $\{\mathbf{v}_{N+1}, \ldots, \mathbf{v}_{N+K}\}$ with small transverse momentum.

- *Collinear safety.* The model is robust to a collinear splitting of a particle, which is represented by replacing a particle $\mathbf{v}_j \in \mathbf{e}$ with two particles $\mathbf{v}_{j_1}$ and $\mathbf{v}_{j_2}$, such that $\mathbf{v}_j = \mathbf{v}_{j_1} + \mathbf{v}_{j_2}$ and $\mathbf{v}_{j_1} \cdot \mathbf{v}_{j_2} = ||\mathbf{v}_{j_1}||\,||\mathbf{v}_{j_2}|| - \epsilon$.

The sequential recombination algorithms lead to an IRC-safe definition of jets, in the sense that given the event $\mathbf{e}$, the number of jets M and their 4-momenta $\mathbf{v}(\mathbf{t}_j)$ are IRC-safe.

An early motivation of this work is that basing the RNN topology on the sequential recombination algorithms would provide an avenue to machine learning classifiers with some theoretical guarantee of IRC safety. If one only wants to ensure robustness to only one soft particle or one collinear split, this could be satisfied by simply running a single iteration of the jet algorithm as a pre-processing step. However, it is difficult to ensure a more general notion of IRC safety on the embedding due to the non-linearities in the network. Nevertheless, we can explicitly test the robustness of the embedding or the subsequent classifier to the addition of soft particles or collinear splits to the input 4-momenta.

Table II shows the results of a non-gated RNN trained on the nominal particle-level input when applied to testing data with additional soft particles or collinear splits. The collinear splits were uniform in the momentum fraction and maintained the small invariant mass of the hadrons. We considered one or ten collinear splits on both random particles and the highest p_T particles. We see that while the 30 models trained with a descending p_T topology very slightly outperform the k_t topology for almost scenarios, their performance in terms of $R_{\epsilon=50\%}$ decreases relatively more rapidly when collinear splits are applied (see e.g., the *collinear10-max* scenarios where the performance of k_t decreases by 4%, while the performance of p_T decreases by 10%). This suggests a higher robustness towards collinear splits for recursive networks based on k_t topologies.

We also point out that the training of these networks is based solely on the classification loss for the nominal sample. If we are truly concerned with the IRC-safety considerations, then it is natural to augment the training of the classifiers to be robust to these variations. A number of modified training procedures exist, including e.g., the adversarial training procedure described in Ref. [27].

TABLE II. Performance of pre-trained RNN classifiers (without gating) applied to nominal and modified particle inputs. The *collinear1* (*collinear10*) scenarios correspond to applying collinear splits to one (ten) random particles within the jet. The *collinear1-max* (*collinear10-max*) scenarios correspond to applying collinear splits to the highest p_T (ten highest p_T) particles in the jet. The *soft* scenario corresponds to adding 200 particles with $p_T = 10^{-5}$ GeV uniformly in $0 < \phi < 2\pi$ and $-5 < \eta < 5$.

Scenario	Architecture	ROC AUC	$R_{\epsilon=50\%}$
nominal	k_t	0.9185 ± 0.0006	68.3 ± 1.8
nominal	desc-p_T	0.9189 ± 0.0009	70.4 ± 3.6
collinear1	k_t	0.9183 ± 0.0006	68.7 ± 2.0
collinear1	desc-p_T	0.9188 ± 0.0010	70.7 ± 4.0
collinear10	k_t	0.9174 ± 0.0006	67.5 ± 2.6
collinear10	desc-p_T	0.9178 ± 0.0011	67.9 ± 4.3
collinear1-max	k_t	0.9184 ± 0.0006	68.5 ± 2.8
collinear1-max	desc-p_T	0.9191 ± 0.0010	72.4 ± 4.3
collinear10-max	k_t	0.9159 ± 0.0009	65.7 ± 2.7
collinear10-max	desc-p_T	0.9140 ± 0.0016	63.5 ± 5.2
soft	k_t	0.9179 ± 0.0006	68.2 ± 2.3
soft	desc-p_T	0.9188 ± 0.0009	70.2 ± 3.7

TABLE III. Summary of event classification performance. Best results are achieved through nested recurrence over the jets and over their constituents, as motivated by QCD.

Input	ROC AUC	$R_{\epsilon=80\%}$
Hardest jet		
$\mathbf{v}(\mathbf{t}_j)$	0.8909 ± 0.0007	5.6 ± 0.0
$\mathbf{v}(\mathbf{t}_j)$, $\mathbf{h}_j^{\text{jet}(k_t)}$	**0.9602 ± 0.0004**	**26.7 ± 0.7**
$\mathbf{v}(\mathbf{t}_j)$, $\mathbf{h}_j^{\text{jet}(\text{desc}-p_T)}$	0.9594 ± 0.0010	25.6 ± 1.4
2 hardest jets		
$\mathbf{v}(\mathbf{t}_j)$	0.9606 ± 0.0011	21.1 ± 1.1
$\mathbf{v}(\mathbf{t}_j)$, $\mathbf{h}_j^{\text{jet}(k_t)}$	0.9866 ± 0.0007	156.9 ± 14.8
$\mathbf{v}(\mathbf{t}_j)$, $\mathbf{h}_j^{\text{jet}(\text{desc}-p_T)}$	**0.9875 ± 0.0006**	**174.5 ± 14.0**
5 hardest jets		
$\mathbf{v}(\mathbf{t}_j)$	0.9576 ± 0.0019	20.3 ± 0.9
$\mathbf{v}(\mathbf{t}_j)$, $\mathbf{h}_j^{\text{jet}(k_t)}$	0.9867 ± 0.0004	152.8 ± 10.4
$\mathbf{v}(\mathbf{t}_j)$, $\mathbf{h}_j^{\text{jet}(\text{desc}-p_T)}$	**0.9872 ± 0.0003**	**167.8 ± 9.5**
No jet clustering, desc-p_T on $\mathbf{v}_i$		
$i = 1$	0.6501 ± 0.0023	1.7 ± 0.0
$i = 1, \ldots, 50$	**0.8925 ± 0.0079**	**5.6 ± 0.5**
$i = 1, \ldots, 100$	0.8781 ± 0.0180	4.9 ± 0.6
$i = 1, \ldots, 200$	0.8846 ± 0.0091	5.2 ± 0.5
$i = 1, \ldots, 400$	0.8780 ± 0.0132	4.9 ± 0.5

VI. EXPERIMENTS WITH EVENT-LEVEL CLASSIFICATION

As in the previous section, we carried out a number of performance studies. However, our goal is mainly to demonstrate the relevance and scalability of the QCD-motivated approach we propose, rather than making a statement about the physics reach of the signal process. Results are discussed considering the idealized *particles* scenario, where the 4-momenta come from perfectly measured stable hadrons from `PYTHIA`. Experiments for the *towers* scenario (omitted here) reveal similar qualitative conclusions, though performance was slightly worse for all models, as expected.

f. Number of jets The first factor we varied was the maximum number of jets in the sequence of embeddings given as input to the GRU. While the event-level embed-

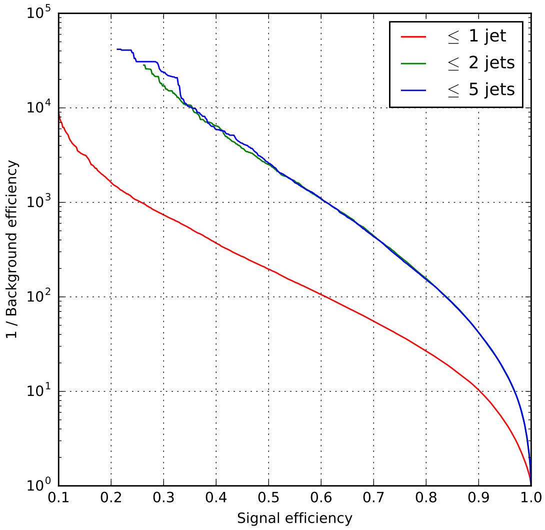

FIG. 5. Event classification performance of the RNN classifier when varying the maximum number of jets given as input to the GRU. This plots shows there is significant improvement from going to the hardest to the 2 hardest jets, while there is no to little gain in considering more jets.

ding can be computed over all the jets it is constituted by, QCD suggests that the 2 highest p_T jets hold most of the information to separate signal from background events, with only marginal discriminating information left in the subsequent jets. As Table III and Fig. 5 show, there is indeed significant improvement in going from the hardest jet to the 2 hardest jets, while there is no to little gain in considering more jets. Let us also emphasize that the event-level models have only 18,681 parameters, and were trained on 100,000 training examples.

g. Topology of the binary trees The second factor we studied was the architecture of the networks used for the inner embedding of the jets, for which we compare k_t against descending p_T topologies. As in the previous section, best results are achieved with descending p_T topologies, though the difference is only marginal.

h. Other variants Finally, we also compare with baselines. With respect to an event-level embedding computed only from the 4-momenta $\mathbf{v}(\mathbf{t}_j)$ (for $j = 1, \ldots, M$) of the jets, we find that augmenting the input to the GRU with jet-level embeddings yields significant improvement, e.g. improving ROC AUC from 0.9606 to 0.9875 (resp. $R_{\epsilon=80\%}$ from 21.1 to 174.5) when considering the 2 hardest jets case. This suggests that jet substructures are important to separate signal from background events, and correctly learned when nesting embeddings. Similarly, we observe that directly feeding the GRU with the 4-momenta $\mathbf{v}_i$, for $i = 1, \ldots, N$, of the constituents of the event performs significantly worse. While performance remains decent (e.g., with a ROC AUC of 0.8925 when feeding the 50 4-momenta with largest p_T), this suggests that the recurrent network fails to leverage some of the relevant information, which is otherwise easier to identify and learn when inputs to the GRU come directly grouped as jets, themselves structured as trees. In contrast to our previous results for jet-level experiments, this last comparison underlines the fact that integrating domain knowledge by structuring the network topology is in some cases crucial for performance.

Overall, this study shows that event embeddings inspired from QCD and produced by nested recurrence, over the jets and over their constituents, is a promising avenue for building effective machine learning models. To our knowledge, this is the first classifier operating at the hadron-level for all the particles in an event, in a way motivated in its structure by QCD.

VII. RELATED WORK

Neural networks in particle physics have a long history. They have been used in the past for many tasks, including early work on quark-gluon discrimination [28, 29], particle identification [30], Higgs tagging [31] or track identification [32]. In most of these, neural networks appear as shallow multi-layer perceptrons where input features were designed by experts to incorporate domain knowledge. More recently, the success of deep convolutional networks has triggered a new body of work in jet physics, shifting the paradigm from engineering input features to learning them automatically from raw data, e.g., as in these works treating jets as images [1–8]. Our work builds instead upon an analogy between QCD and natural languages, hence complementing the set of algorithms for jet physics with techniques initially developed for natural language processing [9–14]. In addition, our approach does not delegate the full modeling task to the machine. It allows to incorporate domain knowledge in terms of the network architecture, specifically by structuring the recursion stack for the embedding directly from QCD-inspired jet algorithms (see Sec. III)

VIII. CONCLUSIONS

Building upon an analogy between QCD and natural languages, we have presented in this work a novel class of recursive neural networks for jet physics that are derived from sequential recombination jet algorithms. Our experiments have revealed that preprocessing steps applied to jet images during their construction (specifically the pixelisation) loses information, which impacts classification performance. By contrast, our recursive network is able to work directly with the four-momenta of a variable-length set of particles, without the loss of information due to discretization into pixels. Our experiments indicate that this results in significant gains in terms of accuracy and data efficiency with respect to previous image-based networks. Finally, we also showed for the first time a hierarchical, event-level classification model operating on all the hadrons of an event. Notably, our results showed that incorporating domain knowledge derived from jet algorithms and encapsulated in terms of the network architecture led to improved classification performance.

While we initially expected recursive networks operating on jet recombination trees to outperform simpler p_T-ordered architectures, our results still clearly indicate that the topology has an effect on the final performance of the classifier. However, our initial studies indicate that architectures based on jet trees are more robust to infrared radiation and collinear splittings than the simpler p_T-ordered architectures, which may outweigh what at face value appears to be a small loss in performance. Accordingly, it would be natural to include robustness to pileup, infrared radiation, and collinear splittings directly in the training procedure [27]. Moreover, it is compelling to think of generalizations in which the optimization would include the topology used for the embedding as learnable component instead of considering it fixed a priori. An immediate challenge of this approach is that a discontinuous change in the topology (e.g., from varying α or R) makes the loss non-differentiable and rules out standard back propagation optimization algorithms. Nevertheless, solutions for learning composition orders have recently been proposed in NLP, using either explicit supervision [33] or reinforcement learning [34]; both of which could certainly be adapted to jet embeddings. Another promising generalization is to use a graph-convolutional network that operates on a graph where the vertices correspond to particle 4-momenta $\mathbf{v}_i$ and the edge weights are given by $d^{\alpha}_{ii'}$ or a similar QCD-motivated quantity [35–41]. In conclusion, we feel confident that there is great potential in hybrid techniques like this that incorporate physics knowledge and leverage the power of machine learning.

ACKNOWLEDGMENTS

We would like to thank the authors of Ref.[6] for sharing the data used in their studies and Noel Dawe in particular for his responsiveness in clarifying details about their work. We would also like to thank Joan Bruna for enlightening discussions about graph-convolutional networks. Cranmer and Louppe are both supported through NSF ACI-1450310, additionally Cranmer and Becot are supported through PHY-1505463 and PHY-1205376.

[1] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwarztman. Jet-Images: Computer Vision Inspired Techniques for Jet Tagging. *JHEP*, 02:118, 2015, 1407.5675.

[2] Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-Images – Deep Learning Edition. 2015, 1511.05190.

[3] Leandro G. Almeida, Mihailo Backović, Mathieu Cliche, Seung J. Lee, and Maxim Perelstein. Playing Tag with ANN: Boosted Top Identification with Pattern Recognition. *JHEP*, 07:086, 2015, 1501.05968.

[4] Pierre Baldi, Kevin Bauer, Clara Eng, Peter Sadowski, and Daniel Whiteson. Jet Substructure Classification in High-Energy Physics with Deep Neural Networks. 2016, 1603.09349.

[5] Daniel Guest, Julian Collado, Pierre Baldi, Shih-Chieh Hsu, Gregor Urban, and Daniel Whiteson. Jet Flavor Classification in High-Energy Physics with Deep Neural Networks. *Phys. Rev.*, D94(11):112002, 2016, 1607.08633.

[6] James Barnard, Edmund Noel Dawe, Matthew J. Dolan, and Nina Rajcic. Parton Shower Uncertainties in Jet Substructure Analyses with Deep Neural Networks. 2016, 1609.00607.

[7] Patrick T. Komiske, Eric M. Metodiev, and Matthew D. Schwartz. Deep learning in color: towards automated quark/gluon jet discrimination. *JHEP*, 01:110, 2017, 1612.01551.

[8] Gregor Kasieczka, Tilman Plehn, Michael Russell, and Torben Schell. Deep-learning Top Taggers or The End of QCD? 2017, 1701.08784.

[9] Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.

[10] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[11] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[12] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. Sentence modeling with gated recursive neural network. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 793–798, 2015.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, chapter 10. MIT Press, 2016. `http://www.deeplearningbook.org`.

[16] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. The Anti-k(t) jet clustering algorithm. *JHEP*, 04:063, 2008, 0802.1189.

[17] Gavin P. Salam. Towards Jetography. *Eur. Phys. J.*, C67:637–686, 2010, 0906.1833.

[18] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[19] Nadine Fischer, Stefan Prestel, Mathias Ritzmann, and Peter Skands. Vincia for Hadron Colliders. 2016, 1605.06142.

[20] M. Ritzmann, D. A. Kosower, and P. Skands. Antenna Showers with Hadronic Initial States. *Phys. Lett.*, B718:1345–1350, 2013, 1210.6345.

[21] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Jesse Thaler and Ken Van Tilburg. Identifying Boosted Objects with N-subjettiness. *JHEP*, 03:015, 2011, 1011.2268.

[24] Samuel R Bowman, Christopher D Manning, and Christopher Potts. Tree-structured composition in neural networks without tree-structured architectures. *arXiv preprint arXiv:1506.04834*, 2015.

[25] Samuel Ryan Bowman. *Modeling natural language semantics in learned representations*. PhD thesis, STANFORD UNIVERSITY, 2016.

[26] Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural mt learn source syntax? In *Proc. of EMNLP*, 2016.

[27] Gilles Louppe, Michael Kagan, and Kyle Cranmer. Learning to Pivot with Adversarial Networks. 2016, 1611.01046.

[28] Leif Lonnblad, Carsten Peterson, and Thorsteinn Rognvaldsson. Finding Gluon Jets With a Neural Trigger. *Phys. Rev. Lett.*, 65:1321–1324, 1990.

[29] Leif Lonnblad, Carsten Peterson, and Thorsteinn Rognvaldsson. Using neural networks to identify jets. *Nucl. Phys.*, B349:675–702, 1991.

[30] R. Sinkus and T. Voss. Particle identification with neural networks using a rotational invariant moment represenetation. *Nucl. Instrum. Meth.*, A391:360–368, 1997.

[31] P. Chiappetta, P. Colangelo, P. De Felice, G. Nardulli, and G. Pasquariello. Higgs search by neural networks at LHC. *Phys. Lett.*, B322:219–223, 1994, hep-ph/9401343.

[32] Bruce H. Denby. Neural Networks and Cellular Automata in Experimental High-energy Physics. *Comput. Phys. Commun.*, 49:429–448, 1988.

[33] Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*, 2016.

[34] Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*, 2016.

[35] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[36] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

[37] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. *CoRR*, abs/1511.05493, 2015.

[38] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. *CoRR*, abs/1605.05273, 2016.

[39] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375, 2016.

[40] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[41] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[42] Dougal Maclaurin, David Duvenaud, Matthew Johnson, and Ryan P. Adams. Autograd: Reverse-mode differentiation of native Python, 2015.

Appendix A: Gated recursive jet embedding

The recursive activation proposed in Sec. III A suffers from two critical issues. First, it assumes that left-child, right-child and local node information $\mathbf{h}^{\text{jet}}_{k_L}$, $\mathbf{h}^{\text{jet}}_{k_R}$, $\mathbf{u}_k$ are all equally relevant for computing the new activation, while only some of this information may be needed and selected. Second, it forces information to pass through several levels of non-linearities and does not allow to propagate unchanged from leaves to root. Addressing these issues and generalizing from [12–14], we recursively define a recursive activation equipped with reset and update gates as follows:

$$\mathbf{h}^{\text{jet}}_k = \begin{cases} \mathbf{u}_k & \text{if } k \text{ is a leaf} \\ \mathbf{z}_H \odot \tilde{\mathbf{h}}^{\text{jet}}_k + \mathbf{z}_L \odot \mathbf{h}^{\text{jet}}_{k_L} + & \text{otherwise} \\ \hookrightarrow \mathbf{z}_R \odot \mathbf{h}^{\text{jet}}_{k_R} + \mathbf{z}_N \odot \mathbf{u}_k & \end{cases} \tag{A1}$$

$$\mathbf{u}_k = \sigma\left(W_u g(\mathbf{o}_k) + b_u\right) \tag{A2}$$

$$\mathbf{o}_k = \begin{cases} \mathbf{v}_{i(k)} & \text{if } k \text{ is a leaf} \\ \mathbf{o}_{k_L} + \mathbf{o}_{k_R} & \text{otherwise} \end{cases} \tag{A3}$$

$$\tilde{\mathbf{h}}^{\text{jet}}_k = \sigma\left(W_{\tilde{h}} \begin{bmatrix} \mathbf{r}_L \odot \mathbf{h}^{\text{jet}}_{k_L} \\ \mathbf{r}_R \odot \mathbf{h}^{\text{jet}}_{k_R} \\ \mathbf{r}_N \odot \mathbf{u}_k \end{bmatrix} + b_{\tilde{h}} \right) \tag{A4}$$

$$\begin{bmatrix} \mathbf{z}_H \\ \mathbf{z}_L \\ \mathbf{z}_R \\ \mathbf{z}_N \end{bmatrix} = \text{softmax}\left(W_z \begin{bmatrix} \tilde{\mathbf{h}}^{\text{jet}}_k \\ \mathbf{h}^{\text{jet}}_{k_L} \\ \mathbf{h}^{\text{jet}}_{k_R} \\ \mathbf{u}_k \end{bmatrix} + b_z \right) \tag{A5}$$

$$\begin{bmatrix} \mathbf{r}_L \\ \mathbf{r}_R \\ \mathbf{r}_N \end{bmatrix} = \text{sigmoid}\left(W_r \begin{bmatrix} \mathbf{h}^{\text{jet}}_{k_L} \\ \mathbf{h}^{\text{jet}}_{k_R} \\ \mathbf{u}_k \end{bmatrix} + b_r \right) \tag{A6}$$

where $W_{\tilde{h}} \in \mathbb{R}^{q \times 3q}$, $b_{\tilde{h}} \in \mathbb{R}^q$, $W_z \in \mathbb{R}^{q \times 4q}$, $b_z \in \mathbb{R}^q$, $W_r \in \mathbb{R}^{q \times 3q}$, $b_r \in \mathbb{R}^q$, $W_u \in \mathbb{R}^{q \times 4}$ and $b_u \in \mathbb{R}^q$ form together the shared parameters to be learned, σ is the ReLU activation function and $\odot$ denotes the element-wise multiplication.

Intuitively, the reset gates $\mathbf{r}_L$, $\mathbf{r}_R$ and $\mathbf{r}_N$ control how to actively select and then merge the left-child embedding $\mathbf{h}^{\text{jet}}_{k_L}$, the right-child embedding $\mathbf{h}^{\text{jet}}_{k_R}$ and the local node information $\mathbf{u}_k$ to form a new candidate activation $\tilde{\mathbf{h}}^{\text{jet}}_k$. The final embedding $\mathbf{h}^{\text{jet}}_k$ can then be regarded as a choice among the candidate activation, the left-child embedding, the right-child embedding and the local node information, as controlled by the update gates $\mathbf{z}_H$, $\mathbf{z}_L$, $\mathbf{z}_R$ and $\mathbf{z}_N$. Finally, let us note that the proposed gated recursive embedding is a generalization of Section III A, in the sense that the later corresponds to the case where update gates are set to $\mathbf{z}_H = 1$, $\mathbf{z}_L = 0$, $\mathbf{z}_R = 0$ and $\mathbf{z}_N = 0$ and reset gates to $\mathbf{r}_L = 1$, $\mathbf{r}_R = 1$ and $\mathbf{r}_N = 1$ for all nodes k.

Appendix B: Gated recurrent event embedding

In this section, we formally define the gated recurrent event embedding introduced in Sec. III B. Our event embedding function is a GRU [21] operating on the p_T ordered sequence of pairs $(\mathbf{v}(\mathbf{t}_j), \mathbf{h}^{\text{jet}}_1(\mathbf{t}_j))$, for $j = 1, \dots, M$, where $\mathbf{v}(\mathbf{t}_j)$ is the unprocessed 4-momentum (ϕ, η, p_T, m) of the jet $\mathbf{t}_j$ and $\mathbf{h}^{\text{jet}}_1(\mathbf{t}_j)$ is its embedding. Its final output $\mathbf{h}^{\text{event}}_{j=M}$ is recursively defined as follows:

$$\mathbf{h}^{\text{event}}_j = \mathbf{z}_j \odot \mathbf{h}^{\text{event}}_{j-1} + (1 - \mathbf{z}_j) \odot \tilde{\mathbf{h}}^{\text{event}}_j \tag{B1}$$

$$\tilde{\mathbf{h}}^{\text{event}}_j = \sigma\left(W_{hx}\mathbf{x}_j + W_{hh}(\mathbf{r}_j \odot \mathbf{h}^{\text{event}}_{j-1}) + b_h\right) \tag{B2}$$

$$\mathbf{x}_j = \begin{bmatrix} \mathbf{v}(\mathbf{t}_j) \\ \mathbf{h}^{\text{jet}}_1(\mathbf{t}_j) \end{bmatrix} \tag{B3}$$

$$\mathbf{z}_j = \text{sigmoid}\left(W_{zx}\mathbf{x}_j + W_{zh}\mathbf{h}^{\text{event}}_{j-1} + b_z\right) \tag{B4}$$

$$\mathbf{r}_j = \text{sigmoid}\left(W_{rx}\mathbf{x}_j + W_{rh}\mathbf{h}^{\text{event}}_{j-1} + b_r\right) \tag{B5}$$

where $W_{hx} \in \mathbb{R}^{r\times 4+q}$, $W_{hh} \in \mathbb{R}^{r\times r}$, $b_h \in \mathbb{R}^r$, $W_{rx} \in \mathbb{R}^{r\times 4+q}$, $W_{rh} \in \mathbb{R}^{r\times r}$, $b_r \in \mathbb{R}^r$, $W_{zx} \in \mathbb{R}^{r\times 4+q}$, $W_{zh}\mathbb{R}^{r\times r}$ and $b_z \in \mathbb{R}^r$ are the parameters of the embedding function, r is the size of the embedding, σ is the ReLU activation function, and $\mathbf{h}^{\text{event}}_0 = 0$. In the experiments of Sec. VI, only the 1, 2 or 5 hardest jets are considered in the sequence $j = 1, \dots, M$, as ordered by ascending values of p_T.

Appendix C: Implementation details

While tree-structured networks appear to be a principled choice in natural language processing, they often have been overlooked in favor of sequence-based networks on the account of their technical incompatibility with batch computation [33]. Because tree-structured networks use a different topology for each example, batching is indeed often impossible in standard implementations, which prevents them from being trained efficiently on large datasets. For this reason, the recursive jet embedding we introduced in Sec. III A would undergo the same technical issues if not implemented with caution.

In our implementation, we achieve batch computation by noticing that activations from a same level in a recursive binary tree can be performed all at once, provided all necessary computations from the deeper levels have already been performed. This principle extends to the synchronized computation across multiple trees, which enables the batch computation of our jet embeddings across many events. More specifically, the computation of jet embeddings is preceded by a traversal of the recursion trees for all jets in the batch, and whose purpose is to unroll and regroup computations by their level of recursion. Embeddings are then reconstructed level-wise in batch, in a bottom-up fashion, starting from the deepest level of recursion across all trees.

Finally, learning is carried out through gradients obtained by the automatic differentiation of the full model chain on a batch of events (i.e., the recursive computation of jet embeddings, the sequence-based recurrence to form the event embeddings, and the forward pass through the classifier). The implementation is written in native Python code and makes use of Autograd [42] for the easy derivation of the gradients over dynamic structures. Code is available at [2] under BSD license for further technical details.

[2] `https://github.com/glouppe/recnn`