The study of the substructure of collimated particles from quarks and gluons, or jets, has the promise to reveal the details how color charges interact with the QCD plasma medium created in colliders such as RHIC and the LHC. Traditional jet substructure observables have been constructed using expert knowledge, and are largely transplanted, unmodified, from the high-energy physics, where the goal is primarily the study of boosted hadronic decays. A novel neural network architecture is described that is capable of examining theoretical models, and constructs, on its own, an analysis procedure that is sensitive to the internal model features. This architecture, in combination with symbolic regression, further allows the extraction of closed-form algebraic expressions from the learned result—enabling the automatically constructed jet substructure analysis to be subsequently understood and reproduced by humans. This system is then tasked to construct an analysis that infers the plasma temperature from observing jets, which is demonstrated using both JEWEL and the Linearized Boltzmann Transport model, and at the presence of a realistic remnant of the plasma, or underlying event, that the measurement has to overcome. In a demonstration how algorithms can produce original research in direct competition to human experts, the resulting jet substructure variables and analyses are capable of determining the initial temperature of the plasma medium from analyzing 1200–2500 jets, a performance not seen in existing, manually designed analyses. Comparison of an incidentally discovered observable with the existing literature further indicates that the system described is capable of examining the model phase spaces to a detail at least comparable to the current field of human experts.

Colliders, such as the Large Hadron Collider (LHC) at CERN, can be used to heat nuclei to very high temperature and compress them to densities many times that of normal nuclei. It has been shown that heavy ion collisions at both RHIC and the LHC undergo a phase transition from normal, bound hadronic matter to a plasma of quarks and gluons. This quark–gluon plasma has surprising properties: it flows as a nearly frictionless fluid, and exhibits a large opacity to transiting quarks and gluons [1].

Analogous to the Bethe formula known for the electromagnetic charge and plasma, a key question is the magnitude and mechanism of energy loss by quarks and gluons (partons) passing through quark–gluon plasma, and how the plasma transports the deposited energy. Addressing this experimentally requires observables sensitive to the interaction between partons and the plasma. A novel approach uses jet substructure observables, built from the angular correlation of energies inside the collimated spray of hadrons (known as a jet) that a parton becomes before reaching the detector.

In heavy ion publications utilizing jet substructure (e.g. [2]), the substructure variables re-use those developed to tag boosted objects in high energy physics. So far, most known substructure analyses are moderately sensitive to the presence of a heavy-ion collision vs. the proton–proton baseline, but do not demonstrate a sensitivity to specific heavy ion model features. Consequently, we do not know whether jet substructure can provide as much information about quark gluon plasma properties as existing measurements of the soft, bulk emission [3]. In this article, examples of novel jet substructure variables are given, together with a neural network (NN) based method that led to their discovery. The demonstration of such an automatically produced, previously unknown result is also a demonstration how algorithms can produce original research in direct competition to human experts.

Possible analyses that can be applied to Quantum Chromodynamics (QCD) in hadron or heavy ion collider experiments can be expressed as a combination of two functions, the per-event observable extraction and a subsequent statistical analysis. A human expert would construct analyses iteratively via generating hypothesis from his or her knowledge or intuition, and testing it against models. However, when the function space of possible analyses is very large, a competing method would be an automated search for an analysis of the desired property, using numerical optimization. Neural Networks are known as universal function approximators [4], and deep layered NN have been demonstrated to be more efficient than traditional “shallow” function approximation techniques [5].

First, the general formulation of analysis functions using the structure of the NN is described. Then, taking advantage of the efficient optimization that can be applied to NN, analyses are constructed by optimizing performance extracting physics parameters from Monte Carlo models, without involving human physics knowledge in designing the analysis. This is in departure from previous instances of automated generation of scientific hypotheses for research, where (non-mathematical) domain-specific, human knowledge are used as input (e.g. [6]).

Events for the NN training are generated for lead–lead (Pb–Pb) collision at a center-of-mass energy of $\sqrt{s_{NN}} = 5.02$ TeV, corresponding to the Large Hadron Collider (LHC) Run-2 data. The impact parameter range sam-
plied corresponds to the 0–10% most central collision geometries among the total inelastic Pb-Pb cross section from the Glauber Monte Carlo in [7]. Jets that interact with the plasma medium are generated using JEWEL 2.2.0 [8] and the Linearized Boltzmann Transport (LBT) model [9]. Events are weighted $\propto p_{\perp}^{5.7}$, where $p_{\perp}$ is the center-of-mass transverse momentum transfer in a single parton-parton scattering. This approximately compensates for the power law decrease of the jet spectrum with jet energy.

JEWEL events are generated with its default plasma model, where the initial time $\tau_i$ is varied between 0.2 and 0.8 fm/$c$, and the mean initial temperature $T_i$ between 0.16 and 0.76 GeV/$k$. LBT events are generated using parton level hard scattering from PYTHIA 8.235 [10] tune CUETP8M1 [11]. Recouling scattering centers are subtracted according to the procedure the JEWEL author has referred to as “4MomSub”, which clusters those medium partons into the jet using near-zero momentum “ghost particles” that are place holders and then subtract their original four-momenta from the jet substructure.

For LBT, in order to study the sensitivity to initial parameters without costly rerunning of numerical hydrodynamics, the temperature and velocity profile is sampled from the viscous Gubser flow [12] as its plasma dynamics model. The integration constant $\tilde{T}_0$ is varied between 0.383 and 0.583 GeV/$k$, such that the resulting medium temperature matches with $T(\tau_i) = T_i$ the same range as JEWEL. As jets in LBT interact with the medium exchange color charges with the recoiling medium partons, the hard scattering event contains color connection with medium partons that are not actually present in the final state. In order to hadronize the event in PYTHIA 8, those medium partons are retained, and are added back to the hard scattering event with zero momenta. The same procedure as “4MomSub” for JEWEL, as described previously, is also applied here.

Neither JEWEL nor LBT produces the particles that is the remnant of the plasma medium, or the underlying event (UE). The UE, for both JEWEL and LBT, is generated using HYDJET 1.9 [13]. Maintaining its default tune, the charged particle multiplicity density $\langle dN_{ch}/d\eta \rangle$ with the track pseudorapidity $|\eta| < 0.5$, at $\sqrt{s_{NN}} = 5.02$ TeV Pb-Pb, is observed to be $\approx 18\%$ higher (2407 ± 5 for 0–2.5% centrality and 1787 ± 4 for 7.5–10% centrality) than experimentally measured [14]. Since the constructed analyses must be robust in a more adverse UE environment than experimentally encountered, the unmodified tune is sufficient for this study. Event centralities are sampled independently in JEWEL and HYDJET, in order to prevent the machine learned analysis to be based on trivial multiplicity effects.

Jets are reconstructed using the anti-$k_T$ algorithm [15] with the distance parameter $D = 0.4$. Jet reconstruction is applied to the hard and UE event final state particles superimposed, thus capturing the effect of imperfect reconstruction due to the presence of the UE. A collision centrality dependent mean UE particle contribution to the jet transverse momentum $\langle p_{T,UE} \rangle$ is determined. This is subtracted to obtain the corrected $p_{T,J} = p_{T,tot} - \langle p_{T,UE} \rangle$ at the hard collision scale. Jets with $100 < p_{T,J} < 500$ GeV/$c$ and $|\eta_J| < 1.4$ are considered for the substructure analysis, ranges that are well-covered by barrel tracking and calorimetry at the LHC. At these energies, jets from hard scattering are reliably distinguished from the combinatorial overlap of bulk-produced particles. The jet spectrum $dN_J/dp_{T,J}$ for different JEWEL and LBT medium scenarios are forced

![FIG. 1. The layout of the statistical analysis learning neural network, where the functions $c$, $s$, and $d$ correspond to the convolutional feature extraction, permutation symmetrization, and dense layers.](image)

![FIG. 2. The upper bound on partial derivative $I_{lm}$, used as regularization, for the per jet–event analysis c that discriminates between JEWEL $T_i$. The index $m \in \{1, 2, \ldots, 489\}$ counts the energy flow polynomial with $1 \leq \deg \leq 7$ (with increasing order), and $l$ is the index of the intermediate analysis. The progression is plotted for three different epochs, being full passes through the training data. Arrows in the last training epoch indicates the variable subsequently found in the approximation term found by symbolic regression (with $l = 1$ and 16).](image)
to be identical, by randomly discarding jets from the scenario with the higher yield. This prevents the NN to produce a non-substructure analysis that measures the jet spectrum. Also, the jets are analyzed as if they are from independent events, since the aim is to observe the effect of the medium on the substructure and not e.g. the momentum balance.

For each jet, the energy flow polynomials (EFP) [16], i.e. intra-jet angular correlation of energies expressed as the product of the relative momentum fraction carried by the final state jet constituents and their relative opening angles, are calculated. The EFP in this article is calculated for particles within a \( \Delta R_{pJ}^d = (\eta - \eta_j)^2 + (\phi - \phi_J)^2 < D^2 \) around the jet axis \( J \), irrespective whether the particle has been clustered into the jet by the jet reconstruction algorithm. This choice accommodates purely calorimetric reconstruction of the jet kinematics, whereas the substructure is then determined from tracking detectors. Each EFP corresponds to a multigraph \( G = (V, E) \), with the vertices \( V \) being \( N_V \) particles that are inside the disk, and edges \( E \) a multiset consisting of pairs of vertices within \( V \). The EFP of \( G \) is

\[
\text{EFP}_G = \sum_{j_1} \cdots \sum_{j_{N_V}} (z_{j_1} \cdots z_{j_{N_V}} \prod_{(k,l) \in E} \theta_{kl}) \tag{1}
\]

where \( z_j = p_{r,j}/p_{r,J} \), and \( \theta_{kl} = \Delta R_{kl}^d = (\eta_k - \eta_l)^2 + (\phi_k - \phi_l)^2 \). The size of the multiset \( E \) is the degree of the polynomial. All 489 primitive polynomials with \( 1 \leq \text{degree} \leq 7 \) are used (note that the only polynomial with degree 0 is the jet \( p_{r,\text{iso}} \)), except for four polynomials with an irreducible rank (4, 4) tensor trace, as they are prohibitively slow to calculate in the presence of \( \approx 400 \) particles inside a jet from central Pb-Pb collision (\( \approx 6 \) minutes per jet on an Intel Haswell Xeon at 3.5 GHz). For efficiency, evaluation of the remaining polynomials has been reimplemented using BLAS [17]. As the handling of UE by the Neural Network is of interest, the jet substructure is not further manually corrected to the hard scale.

As a second possible jet substructure variable, a polynomial expansion of the jet shape in \( \Delta R_{pJ} \), \((\eta - \eta_J)/(\phi - \phi_J)\), and \( z_p \) between the constituent particle \( p \) and jet axis \( J \) was also explored. For an expansion into a comparable number of coefficients, the performance was significantly below that of the EFP.

Fig. 1 shows the schematic layout of the NN employed in this article. In this NN, \( f(x) = d(s(c(x))) \) is a function of the observable \( x \), which is a series expansion of the internal substructure formed by the final state particles of a reconstructed jet. The function \( f \) is composed of the following groups of NN layers:

1. The convolutional observable extraction layers \( c : \mathbb{R}^{K \times N} \to \mathbb{R}^{M \times N} \), where \( N \) is the number of input jets, \( K \) the number of input features/x observables per jet–event, and \( M \) the number of machine-learned observables per jet–event;

2. the symmetrization layer \( s : \mathbb{R}^{M \times N} \to \mathbb{R}^{M \times N} \);

3. the statistical analysis layers \( d : \mathbb{R}^{M \times N} \to \mathbb{R} \).

The presence of \( s \) is needed, because the connection between neurons in a NN are inherently sensitive to the ordering of its input. Placing \( s \) in front of \( d \) however, allows one to “retrofit” an otherwise ordered NN with permutation invariance. \( M = 1 \) would still satisfy the universal approximation theorem, the presence of a small layer creates a choke point inside the NN and potentially deteriorates the convergence property observed in [18].

In a NN, each layer \( k \) is an operation of the form

\[
x_{k+1}(x_k) = \sigma(Wx_k + b) \tag{2}
\]

where the matrix \( x_k \) is the input of the \( k \)-th layer, \( W \) the weight matrix, \( b \) the bias vector, and \( \sigma \) a nonlinear activation function. Each neuron consists of the weight (a matrix multiplied with the input vector), a bias (the vector added after the matrix multiplication) and a suitable activation function. During the course of training, the each neuron learns an “activation”, which is a real number usually close to \( \pm 1 \). Hence, a single neuron without the activation function is a linear classifier. The activation function – termed analogously to the function of the firing rate in a biological neuron – is needed to produce nonlinear classification.

The initial layers \( c \) represent per-jet–event analysis, which are identical functions applied to a single jet–event, and repeated with the \( N \) jet–events. Each layer in \( c \) is therefore a special case of a one-dimensional, discrete convolution.

These convolutions of \( N \) jet–events represent a sizeable statistical sample, so the sample mean and standard deviation can be reliably determined. A batch normalization [19] is made to the \( N \) jet–events in each layer in \( c \), before application of the activation function. In this step, the mean is subtracted from all values; the differences are inversely scaled by the batch standard deviations. This scales the values to be approximately in \([-1, 1]\), and prevents neurons from becoming permanently “stuck”, i.e. neurons which never activate across the entire training dataset. This is also alleviated by making the network sufficiently large to allow for redundancy.

The symmetrization operation \( s \) transforms the \( M \times N \) matrix input \((x_{j\pi(k)})\), \( j \in \{1, \ldots, M\} \), \( k \in \{1, \ldots, N\} \), into polynomials that enforce invariance under the identical permutation \( \pi \) of the columns corresponding to individual jet–events, but breaks under a \((x_{j\pi(k)})\), if two rows exist with different permutations \( \pi_j \). This avoids limiting the NN to \( M = 1 \), and differs from the construction using symmetric polynomials, e.g. employed in [20], which would describe a function space other than that of possible statistical analyses. The polynomials of order \( m = 1 \) are identical to the elementary symmetric polynomials

\[
s_j(x) = \sum_{k=1}^{N} x_{j\pi(k)} \tag{3}
\]
A possible choice for $N = 2$ and $m = 2$ is

\begin{align}
    s_1(x) &= x_1(1)x_2(1) + x_1(2)x_2(2) \\
    s_2(x) &= x_1(1)x_2(2) + x_1(2)x_2(1)
\end{align}

(4)

and for $N \geq 3$ and $m \geq 2$

\begin{align}
    s_f(x) &= \sum_{k=1}^{N} x_1^{m-[m/2]_{k}} x_1^{l} \mod M, \pi(k).
\end{align}

(5)

Finally, $d$ consists of fully connected NN layers, where the final layer outputs into two output neurons, one for each scenario of the heavy-ion medium. The output neurons are trained with the goal of being 1 for the correct medium scenario, and 0 for the incorrect one, but will attain a probability-like value in between, when the discrimination is imperfect.

The activation function has to be nonlinear (or the NN is reducible to a purely linear model), and is chosen to be $\sigma(x_j) = \max(0, x_j)$, the rectified linear unit (ReLU) that was found to be efficiently trainable [21]. The exception is the last layer, where the activation function is the softmax function $\sigma(x_j) = \exp(x_j) / \sum_k \exp(x_k)$ that converts a vector with multiple values into a probabilistic value between 0 and 1 [22].

A NN trained without further constraints is difficult to analyze, because the network’s ability for discrimination is distributed among all possible inputs, without an easy way to determine, a posteriori, whether part of the NN is redundant or not contributing to the overall performance. The NN employed here is regularized during training using a metric $R$ that approximates how many input terms are needed describe the function inside the NN, or the input complexity. For a NN output after the $k$-th layer, the absolute partial derivative from the $m$-th input to the $l$-th output is bounded by

\begin{align}
    I_{lm} &= \max \left\{ \left| \inf \left( \frac{\partial x_{k,l}}{\partial x_{1,m}} \right) \right|, \left| \sup \left( \frac{\partial x_{k,l}}{\partial x_{1,m}} \right) \right| \right\}
\end{align}

(6)

where inf, sup are the lower and upper interval arithmetic bounds. The sum of absolute magnitude, or $\ell^1$ norm of $I_{lm}$, inversely scaled by the maximum value

\begin{align}
    R &= \frac{\sum I_{lm}}{\max I_{lm}}.
\end{align}

(7)

is then an upper bound of the input complexity that is easily calculable during the NN training, and can be used to guide the NN off configurations where an excessive number of input variables are used, such that it becomes difficult to distinguish input essential vs. redundant to the performance of the NN. Similar ideas exist in the literature, like weight decay [23] and the layer-wise regularization by the Lipschitz continuity [24]. Unlike existing regularization in the literature, $R$ targets the input complexity specifically, and does not prevent hidden layers from using multiple neurons to form nonlinear functions.

Fig. 2 shows how $I_{lm}$ for the layer group $c$ evolves, when trained to discriminate $T_i$ in JEWEL. The index $m \in \{1, 2, \ldots, 489\}$ counts the energy flow polynomial with $1 \leq \text{degree} \leq 7$, and $l$ is the index of the interme-
diately analysis that c outputs. The progression is plotted after three different epochs, which are the number of full passes through the training data. Arrows in the last training epoch indicates the variable the symbolic regression are not the highest values of (7) for layer $I_m$ is then applied to NN layer groups of moderate value of $I_m$.

The function for minimization (loss function) by the NN is

$$L = H + \mu_c R_c + \mu_d R_d$$

where $H = -\log(p)$ the cross entropy, with $p$ being the probability from multiple pseudoexperiments that the NN has determined the correct medium scenario from $N$ jet–events. $R_c$ and $R_d$ are the values of (7) for layer groups $c$ and $d$. The constants $\mu_c, \mu_d$ are regularization parameters that must be adjusted to achieve a particular trade-off how many input terms the trained NN will require to achieve its performance (being $H$), vs. $H$ itself. This type of $l^1$-optimization with competing objectives, that are linked together with an adjustable parameter, is also referred to in the literature as the least absolute shrinkage and selection operator (LASSO) [25].

The NN is implemented in TENSORFLOW 1.10.0 [26], running on a Nvidia GP102 at 1.4 GHz with cuDNN 7.1 [27]. The symbolic regression (SR) algorithm FFX [28] is then applied to NN layer groups $c$ and $d$ to extract closed-form expressions that approximates the function of those NN layer groups.

Training in the case of JEWEL is performed on 65 pseudoexperiments, with each having a high and a low $T_i$ (i.e. 130 pseudoexperiments in total). Each pseudoexperiment contains 2500 jet–events (not shared between pseudoexperiments). The Adam optimization algorithm with its default parameters in [29] is used. Whenever the optimization algorithm passes the entire $130 \times 2500$ unique events, the optimization is considered to have completed an optimization epoch. After 1000 epochs using $\mu_c = \mu_d = 0.01$, the accuracy is $0.973 \pm 0.013$ (68% Jeffreys interval).

SR incrementally tests expressions with increasing size complexity, and retains for each size complexity which form of expression achieved the best accuracy. For JEWEL, the overall analysis (extracted using the first of the two output neurons) is approximated by

$$d_{1,SR} = 8.99 - 0.176N\langle c_{16} \rangle - 0.175N\langle c_1 \rangle$$

where $\langle \rangle$ is the arithmetic mean, and $N = 2500$ is the jet–event count per pseudoexperiment during the training. The subscript indicates that the approximation $d_{1,SR} \approx d_1$ was made by the SR. This particular approximation is the highest complex one that does not involve correlation between multiple jet substructure observables, and has a normalized mean square error (NMSE) of 10.9%, vs. 28.3% for a constant.

The activation of $c_{1,SR} \approx c_1$ and $c_{16,SR} \approx c_{16}$ are given by

$$c_{1,SR} = 0.0112 +$$

$$+ 45.5\theta_{ab}\theta_{bc}\theta_{bd}\theta_{ad}\theta_{ac}\theta_{ce}z_a \cdots z_e +$$

$$+ 20.96\theta_{ab}\theta_{bc}\theta_{bd}\theta_{ad}\theta_{ac}\theta_{ce}z_a \cdots z_e +$$

$$+ 17.70\theta_{ab}\theta_{bc}\theta_{bd}\theta_{ad}\theta_{ac}\theta_{ce}z_a \cdots z_e +$$

$$+ 8.636\theta_{ab}\theta_{bc}\theta_{bd}\theta_{ad}\theta_{ac}\theta_{ce}z_a \cdots z_e -$$

$$- 3.086\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e -$$

$$- 1.086\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e +$$

$$+ 0.769\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e -$$

$$- 0.233\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e -$$

$$- 0.03776\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e -$$

$$- 0.00483\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e -$$

$$- 0.000508\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e -$$

$$- 4.51 \times 10^{-5}\theta_{ac}\theta_{ad}\theta_{ac}\theta_{de}z_a \cdots z_e$$

FIG. 4. Dependence of the symbolic regression approximated $c_{1,SR}$ and $c_{16,SR}$ that discriminates $T_i$ in JEWEL, for various tagged number of splitting induced by jet–plasma interaction, normalized by the integral of each. The dashed line represents the distribution of the untagged JEWEL (averaged over $T_i$ and $T_c$ scenarios). Also shown are the distributions from PYTHIA 8.235 tune CUETP8M1 and Herwig 7.1.1 tune H7.1-Default, both embedded into the same Pb-Pb 0–10% underlying event. Both the information on the splitting, and PYTHIA 8/Herwig 7 events, were not made available to the NN.
\[ c_{16,SR} = 0.0362 + 
+ 0.59\theta_{ab}^{2}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} + 
+ 0.575\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} + 
+ 0.421\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} + 
+ 0.420\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} + 
+ 0.246\theta_{ab}^{2}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} + 
+ 0.187\theta_{ab}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} + 
+ 0.120\theta_{ab}^{2}\theta_{ac}\theta_{ad}\theta_{ae}z_{a}\cdots z_{e} - 
- 0.0465\theta_{ab}^{3}z_{a}z_{b} - 0.0453\theta_{ab}^{2}z_{a}z_{b} - 
- 0.0336\theta_{ab}^{3}z_{a}z_{b} - 0.0328\theta_{ab}^{2}z_{a}z_{b} - 
- 0.0196\theta_{ab}^{3}z_{a}z_{b} - 0.0146\theta_{ab}z_{a}z_{b} - 
- 0.00963\theta_{ab}^{3}z_{a}z_{b} \] 

where the Einstein summation is implied over \( a, b, \ldots, h \), 
\( z_{a}z_{b}z_{c}z_{d} \) has been shortened into \( z_{a}\cdots z_{c} \), and 
\( z_{a}\cdots z_{c} \) as \( z_{a} \) into \( z_{a} \). 

For LBT, discrimination is far easier to achieve, and 
the sample size can be reduced to 50 pseudoexperiments 
(100 for both \( \bar{T} \) scenarios) containing \( N = 1200 \) jet– 
events each. The regularization at \( \mu_{1} = \mu_{2} = 30 \) 
was able to yield a validation accuracy of 0.975 \pm 0.014, 
and the corresponding SR approximated analysis 
\[ d_{1,SR} = 4.49 - 0.318N(c_{13}) - 0.00653N(c_{13})^2 \] 

(NMSE is 1.17%, vs. 36.9% for a constant) involves a 
single variable, which is approximated by SR as 
\[ c_{13,SR} = 0.0453 - 
- 0.00109\log_{10}(p_{1})(\log_{10}(p_{2}) + \log_{10}(p_{3})) - 
- 0.000829\log_{10}(p_{2})\log_{10}(p_{3}) \] 

where 
\[ p_{1} = \theta_{ab}\theta_{ac}\theta_{bc}\theta_{ad}\theta_{bd}\theta_{bc}\theta_{ae}z_{a}\cdots z_{c} \] 
\[ p_{2} = \theta_{ab}\theta_{ac}\theta_{bc}\theta_{ad}\theta_{be}\theta_{ae}z_{a}\cdots z_{c} \] 
\[ p_{3} = \theta_{ab}\theta_{ad}\theta_{bd}\theta_{cd}\theta_{ae}z_{a}\cdots z_{c} \] 

The \( \log_{10} \) function appears to be selected by SR as a function 
of convenience to represent the range compression 
inside the NN, and is unlikely to have deeper meaning. 

Fig. 3 shows the distribution of \( c_{1,SR} \) and \( c_{16,SR} \) 
for various JEWEL \( T_{i} \), and \( c_{13,SR} \) for LBT and various \( \bar{T} \). 
The \( c_{1,SR} < 0 \) for JEWEL is a particularly striking region, 
where \( T_{i} \) induces a change by an order of magnitude. 

From the form of the expression obtained, one can see 
that angular correlations between five particles are 
frequently used. To investigate this further, JEWEL was 
modified to allow book-keeping of the angular direction 
of the originating parton each time a splitting occurred. 
For each reconstructed jet, the number of splittings 
within the angular extent of the jet is used to count 
splittings due to plasma medium interaction. Additionally, 
the Lund string model based PYTHIA 8.235 tune 
CUETP8M1, and Herwig 7.1.1 tune H7.1-Default [30] 
that is based on the Webber model of cluster fragmentation, 
both embedded into the identical Pb-Pb 0–10% UE 
as for JEWEL and LBT, were included to check for the 
p-p expectation including potential fragmentation model 
dependence.

Fig. 4 shows the distribution of \( c_{1,SR} \) and \( c_{16,SR} \) in 
JEWEL with 0, 1–4, and > 4 splittings. Overlaid are also 
PYTHIA 8 and Herwig 7 embedded into Pb-Pb. The two 
variables exhibit significant dependence on the number of 
plasma-induced splittings, with no significant fragmentation 
function dependence upon comparing PYTHIA 8 to 
Herwig 7. The variable \( c_{16,SR} \) evolves with \( T_{i} \) by tag-
ging jets with 1–4 splittings, while losing resolving power 
thereafter (\( c_{16,SR} \) for those jets resemble the average p-p jet). 
On the other hand, \( c_{1,SR} \) has a region \( c_{1,SR} < 0 \) 
that is additionally sensitive to > 4 splittings.

Interestingly, the same \( c_{1,SR} < 0 \) (a quantile containing 
6.65 \pm 0.04\% of the jets in JEWEL averaged over the \( T_{i} \) 
and \( \bar{T} \) scenarios) is populated due to the recoiling scattering 
centers in JEWEL. Its significance vs. PYTHIA 8 
(0.68 \pm 0.11\% of the jets) and Herwig 7 (1.4 \pm 0.2\% of 
the jets) provides a strong constraint whether the recoil 
effect as suggested by JEWEL exists. The only known 
indication of a similar effect between JEWEL recoil on and 
off is the groomed jet mass, which was recently studied 
by CMS [31]. One should point out the NN learned of 
the JEWEL recoil incidentally, and not by a training pro-
cedure targeting this effect. And unlike human experts, 
the NN was never able to observe JEWEL with either the 
recoil effects switched off, or without the full Pb-Pb UE. 
This hints that the machine learning technique presented 
here, operating on a commodity, off-the-self hardware, is 
at least able to study the characteristics of JEWEL to 
a detail comparable to the human experts in the current 
heavy-ion field.

This article describes a system of machine learning 
for the discovery of jet substructure analyses, using 
a NN structured to automatically learn statistical analyses. 
The approach includes regularization to sufficiently apply 
a simplification process during the training. Symbolic Regression 
is then used to extract properties of the resulting 
NN, and can reduce to compact, closed-form expres-
sions readily repeatable by humans. Applied to JEWEL 
and LBT, two heavy-ion jet MC event generators, it con-
structs analyses that can reliably extract the initial tem-
perature, at the presence of full bulk Underlying Event 
– a performance not previously demonstrated by human 
constructed jet substructure analyses. In JEWEL, the 
type of jet substructure observables being constructed 
by machine learning is found to be strongly dependent 
on, i.e. tagging, the number of splittings experienced by 
a parton transiting a quark gluon plasma medium. One of 
the discovered observables also tags the medium recoil 
from the interaction, which is otherwise only known to 
be measurable via the groomed jet mass. This indicates 
that the machine learning technique described here per-
forms original research with jet substructure observables,
at a level comparable to the human experts in the field.

I would like to thank Barbara Jacak for discussions and critical reading, additionally Xin-Nian Wang and Yayun He for the Linearized Boltzmann Transport model and insightful suggestions.

[1] U. Heinz and R. Snellings, Ann. Rev. Nucl. Part. Sci. 63, 123 (2013); K. Adcox et al., Nucl. Phys. A 757, 184 (2005); J. Adams et al., Nucl. Phys. A 757, 102 (2005).
[2] Y. Mehtar-Tani and K. Tywoniuk, Journal of High Energy Physics 2017, 125 (2017); Y.-T. Chien and I. Vitev, Phys. Rev. Lett. 119, 112301 (2017); R. Kunnawalkam Elayavalli and K. C. Zapp, Journal of High Energy Physics 2017, 141 (2017); A. M. Sirunyan et al. (CMS Collaboration), Phys. Rev. Lett. 120, 142302 (2018).
[3] H. Song, S. A. Bass, U. Heinz, T. Hirano, and C. Shen, Phys. Rev. Lett. 106, 192301 (2011); M. Luzum, Phys. Rev. C 83, 044911 (2011); H. Song, S. A. Bass, and U. Heinz, Phys. Rev. C 83, 054912 (2011); C. Gale, S. Jeon, B. Schenke, P. Tribedy, and R. Venugopalan, Phys. Rev. Lett. 110, 012302 (2013).
[4] K. Hornik, M. Stinchcombe, and H. White, Neural Networks 2, 359 (1989); M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Neural Networks 6, 861 (1993).
[5] H. Mhaskar and T. A. Poggio, “Deep vs. shallow networks: An approximation theory perspective,” (2016), arXiv:1608.03287 [cs.LG]; S. Liang and R. Srikant, “Why deep neural networks for function approximation?” (2016), arXiv:1610.04161 [cs.LG]; D. Yarotsky, Neural Netw. 94, 103 (2017).
[6] J. M. Zytkowski, J. Zhu, and A. Hussam, in Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 2, AAAI’90 (AAAI Press, 1990) pp. 889–894; R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, Artificial Intelligence 61, 209 (1993); R. D. King et al., Nature 427, 247 (2004); J. B. Voytek and B. Voytek, J. Neurosci. Methods 208, 92 (2012); L. R. De Vesine, K. Anderson, M. Zreda, C. Zweck, and L. Bradley, in Discovery Informatics: AI Takes a Science-Centered View on Big Data, AAAI Fall Symposium, Technical Report, Vol. FS-13-01 (AAI Access Foundation, 2013) pp. 16–22; S. Spangler et al., in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14 (ACM, New York, NY, USA, 2014) pp. 1877–1886; A. Kadurin, S. Nikolenko, K. Khrabrov, A. Aliper, and A. Zhavoronkov, Mol. Pharm. 14, 3098 (2017); M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, and A. Zeilinger, Phys. Rev. Lett. 116, 090405 (2016).
[7] B. Abelev et al. (ALICE Collaboration), Phys. Rev. C 88, 044909 (2013).
[8] K. Zapp, Eur. Phys. J. C 74, 2762 (2014).
[9] Y. He, T. Luo, X.-N. Wang, and Y. Zhu, Phys. Rev. C 91, 054908 (2015).
[10] T. Sjostrand et al., Comput. Phys. Commun. 191, 159 (2015).
[11] V. Khachatryan et al., Eur. Phys. J. C 76, 155 (2016).
[12] S. S. Gubser, S. S. Pufu, and A. Yarom, Phys. Rev. D 78, 066014 (2008); S. S. Gubser, Phys. Rev. D 82, 085027 (2010).
[13] I. P. Lokhtin and A. M. Snigirev, Eur. Phys. J. C 45, 211 (2006).
[14] J. Adam et al. (ALICE Collaboration), Phys. Rev. Lett. 116, 223202 (2016).
[15] M. Cacciari, G. P. Salam, and G. Soyez, J. High Energy Phys. 2008, 063 (2008).
[16] P. T. Komiske, E. M. Metodiev, and J. Thaler, Journal of High Energy Physics 2018, 13 (2018).
[17] J. Dongarra, Int. J. High Perform. Comput. Appl. 16, 1 (2002); L. S. Blackford et al., ACM Trans. Math. Softw. 28, 135 (2002).
[18] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun, in Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, Vol. 38, edited by G. Lebanon and S. V. N. Vishwanathan (2015) pp. 192–204; B. D. Haeffele and R. Vidal, “Global optimality in tensor factorization, deep learning, and beyond,” (2015), arXiv:1506.07540 [cs.NA]; M. Janzamin, H. Sedghi, and A. Anandkumar, “Generalization bounds for neural networks through tensor factorization,” (2015), arXiv:1506.08473 [cs.LG].
[19] S. Ioffe and C. Szegedy, in Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research, Vol. 37, edited by F. Bach and D. Blei (2015) pp. 448–456.
[20] B. Jiang and H. Guo, J. Chem. Phys. 139, 054112 (2013).
[21] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, Nature 405, 947 (2000); X. Glorot, A. Bordes, and Y. Bengio, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, Vol. 15, edited by G. Gordon, D. Dunson, and M. Dudík (2011) pp. 315–323.
[22] J. S. Bridle, in Neurocomputing, NATO ASI Series F: Computer and Systems Sciences, Vol. 68, edited by F. F. Soulié and J. Hérault (Springer, Berlin, Heidelberg, 1990) pp. 227–236.
[23] A. Krogh and J. A. Hertz, in Advances in Neural Information Processing Systems 4, edited by J. E. Moody, S. J. Hanson, and R. P. Lippmann (Morgan-Kaufmann, 1992) pp. 950–957.
[24] M. Ciáse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, “Parseval networks: Improving robustness to adversarial examples,” (2017), arXiv:1704.08847 [stat.ML]; H. Gouk, E. Frank, B. Pfahringer, and M. Cree, “Regularisation of neural networks by enforcing Lipschitz continuity,” (2018), arXiv:1804.04368 [stat.ML].
[25] F. Santosa and W. W. Symes, SIAM J. Sci. Stat. Comput. 7, 1307 (1986); R. Tibshirani, J. Royal Stat. Soc. B 58, 267 (1996).
[8] T. McConaghy, “FFX: fast, scalable, deterministic symbolic regression technology,” in *Genetic Programming Theory and Practice IX*, edited by R. Riolo, E. Vladislavleva, and J. H. Moore (Springer, New York, NY, 2011) pp. 235–260; in *2011 IEEE Custom Integrated Circuits Conference (CICC)* (IEEE, Piscataway, NJ, 2011) pp. 1–8.

[29] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” (2014), arXiv:1412.6980 [cs.LG].

[30] M. Bähr et al., *The European Physical Journal C* **58**, 639 (2008); J. Bellm et al., *The European Physical Journal C* **76**, 196 (2016).

[31] A. M. Sirunyan et al. (CMS), (2018), arXiv:1805.05145 [hep-ex].