Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2025, 13(11), 4128-4146
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Machine Learning in High-Energy Physics: Illustrative Applications in Theory and Experiment

Maggie Li[1], Neil Pandya[2]

[1]Jericho High School, Jericho, NY, USA
[2]Stevens Institute of Technology, Hoboken, USA
Email: maggie.li1019@gmail.com

How to cite this paper: Li, M. and Pandya, N. (2025) Machine Learning in High-Energy Physics: Illustrative Applications in Theory and Experiment. *Journal of Applied Mathematics and Physics*, **13**, 4128-4146.
https://doi.org/10.4236/jamp.2025.1311228

Received: October 12, 2025
Accepted: November 24, 2025
Published: November 27, 2025

Abstract

Machine learning (ML) has become an increasingly central component of high-energy physics (HEP), providing computational frameworks to address the growing complexity of theoretical calculations and experimental data at the Large Hadron Collider (LHC) and beyond. This review surveys three illustrative domains in which ML is reshaping HEP workflows: Feynman integral evaluation, charged particle tracking, and jet tagging. On the theoretical side, ML methods support more efficient integration-by-parts (IBP) reductions and approximate differential equation (DE) solutions for master integrals. In detector and classification tasks, models ranging from graph-based networks for charged particle tracking to neural networks for jet tagging demonstrate strong performance and scalability under high-luminosity conditions. Collectively, these efforts highlight the potential for ML to enhance precision, efficiency, and interpretability in particle physics and to make previously intractable problems tractable. Ongoing developments in physics-informed architectures, graph-based learning, and large-model integration may further position ML as a foundational tool for the next generation of research in HEP.

Keywords

Machine Learning, High-Energy Physics, Large Hadron Collider, Feynman Integral Reduction, Physics-Informed Neural Networks, Neural Networks, Charged Particle Tracking, Jet Tagging

1. Introduction

High-energy physics (HEP) studies the most fundamental constituents of matter and the principles that govern their interactions. At the center of research in this

field is the Large Hadron Collider (LHC) at CERN [1], which is the largest and most powerful particle accelerator in the world. At the LHC, two proton beams are accelerated almost to the speed of light and driven to collide at a frequency of 40 million times per second. The LHC enables precision tests of the Standard Model alongside searches for physics beyond it, leading to landmark discoveries such as the Higgs boson in 2012 [2]. The two general-purpose detectors, ATLAS and CMS, capture the aftermath of collisions and generate large volumes of data that can be reconstructed and analyzed with high accuracy.

The magnitude of LHC datasets, combined with the theoretical rigor and experimental demands of HEP, creates computational challenges that strain traditional approaches. Theoretical calculations require the evaluation of integrals that quickly become intractable [3]. Reconstructing charged particle trajectories from dense and noisy data overwhelms conventional tracking algorithms [4]. Classifying jets based on hand-crafted observables is inefficient and unable to capture the richness of their internal structure [5]. As both precision and scale increase, new strategies are required to ensure that theoretical predictions and experimental analyses remain manageable.

Machine learning (ML) has emerged as a transformative tool in this context, offering not only efficient tools but also novel ways of framing and solving physics problems. This review focuses on three domains in which ML has had a significant impact on the theory-experiment interface at the LHC. At the theoretical level, ML improves the evaluation of Feynman integrals [6] by enhancing the seeding of integration-by-parts (IBP) and by approximating solutions to differential equations (DEs). At the detector level, graph-based ML architectures [7] have been applied to the reconstruction of charged particle trajectories, allowing more accurate and scalable tracking than traditional combinatorial filters. Finally, in experimental classification tasks, advanced neural network models have provided new avenues for jet tagging [8].

Together, these developments illustrate how ML is transforming both the theoretical and experimental practices of HEP, bridging the gap between them and providing computational tools that are increasingly indispensable for advancing particle physics research. Of course, ML methods are still relatively underdeveloped in physics compared to fields like image recognition and natural language processing, where architectures have matured over decades. In HEP, much more remains to be developed, from adapting algorithms for unique detector data to building models robust enough to aid discovery. This review highlights a selected set of applications in HEP, while numerous others remain outside its scope.

The remainder of this review is organized as follows. Section 2 introduces relevant ML architectures. Sections 3 - 5 examine their applications in HEP, focusing on Feynman integral evaluation, charged particle tracking, and jet tagging, respectively. Section 6 concludes with a discussion of future prospects.

2. Machine Learning Background

The idea of machines that could learn or reason like humans has long fascinated

scientists and writers. The intellectual foundations of ML began to take shape when researchers such as Alan Turing [9] proposed formal ways of thinking about computation and intelligence. Although progress slowed at times, the field was revitalized by advances in computing hardware, the rise of large datasets, and the development of more effective training algorithms.

Today, ML and broader artificial intelligence (AI) are widely applied in scientific fields such as robotics, agriculture, biology, and medicine. Generative AI [10], in particular, has recently surged in popularity and shows potential in physics to accelerate simulations, aid data analysis, guide experimental design, and support theoretical calculations. In HEP, one of the main drivers of progress is neural networks [8]. A neural network [11] is a network of interconnected elements that mimics the structure of the human brain, composed of biological neurons. These characteristics allow it to produce an output pattern when given an input pattern.

The core of modern ML is the concept of deep learning based on deep neural networks (DNNs) [12], which is a feedforward neural network composed of an input layer, multiple hidden layers of interconnected neurons, and an output layer. As shown in **Figure 1**, each hidden layer takes the weighted signals from the previous layer, applies an activation function, and passes the transformed values forward. These layers are hidden because they are not directly exposed to the input or output. Rather, they learn internal representations that capture patterns in the data.

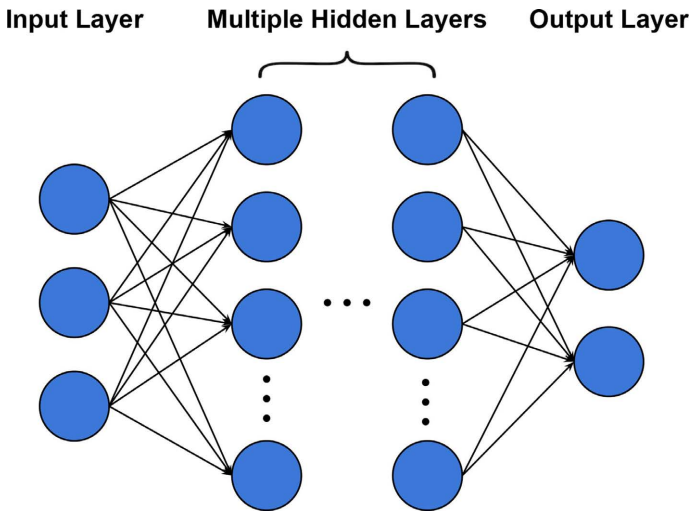

Figure 1. Simple representation of a DNN with an input layer, multiple hidden layers, and an output layer. Each circle represents a neuron. The lines denote weighted connections that transmit signals between neurons, and the dots indicate that many more hidden layers can be stacked in between. The output layer produces the final result of the network.

Convolutional neural networks (CNNs) [13] introduce convolutional layers that apply learnable filters (small matrices of weights that slide across the input and adjust during training to detect useful local patterns) to extract local features from the input data. This design makes CNNs particularly effective for recogniz-

ing patterns in spatially structured data, such as images. By encoding spatial location directly into the architecture, CNNs have become the dominant approach in image recognition and computer vision.

While CNNs handle spatial structure, recurrent neural networks (RNNs) [14] address sequential data by introducing cyclic connections that allow information to be fed back into the network. Such data could include handwriting, genomes, and text. A widely used variant is the Long Short-Term Memory (LSTM) [15] network, which captures long-range dependencies more effectively than standard RNNs by maintaining information over extended sequences.

Graph neural networks (GNNs) [16] generalize neural networks to data represented as graphs, where entities are expressed as nodes connected by edges. Traditional neural networks like CNNs and RNNs are limited to Euclidean data (grids or sequences), whereas GNNs can operate directly on non-Euclidean graph structures. This makes GNNs effective for tasks such as node classification, clustering, and link prediction.

Transformers [17] replace recurrence with the so-called self-attention mechanism, which allows each element of a sequence to directly attend to all others. This allows long-range dependencies to be modeled in parallel. Built with stacked encoder-decoder layers of multi-head attention and feedforward networks, they achieve greater efficiency and scalability than recurrent or convolutional sequence models.

Physics-informed deep learning (PIDL) [18] trains neural networks to solve supervised tasks while enforcing physical laws expressed as nonlinear partial DEs. Using automatic differentiation to embed conservation principles and symmetries, these models enable data-efficient solutions of partial DEs, discovery of governing equations, and construction of physics-informed surrogate models.

Together, these architectures and methods exemplify key components of the modern ML landscape and underpin many recent advances in HEP. Each framework offers unique strengths and capabilities to handle spatial, sequential, relational, or physics-informed data. The following sections examine how these architectures have been applied to three central areas of HEP: Feynman integral evaluation, charged particle tracking, and jet tagging.

3. ML for Feynman Integral Evaluation

The Standard Model of particle physics [19]-[21] is the best-established framework for describing the fundamental particles and three of the four known forces of nature, excluding gravity. To predict how these particles behave and interact, physicists rely on quantum field theory. Feynman integrals [22] provide the mathematical framework for predicting particle interactions, appearing ubiquitously in quantum field theory through perturbative calculations of scattering amplitudes and correlation functions. Accurately evaluating them is essential for connecting theory with experiment, as they play a central role in predicting the outcomes of complex collision processes at the LHC. However, Feynman integrals involve com-

plex, high-dimensional functions and, as a result, are computationally intensive. Recent developments have integrated ML techniques into the two vital stages of Feynman integral evaluation, namely reducing integrals by IBP and solving the resulting integrals through DEs. These techniques have improved the efficiency of Feynman integral evaluations that traditionally require computationally intensive methods.

For each Feynman integral, there is generally an associated graph (a Feynman diagram) with a certain number of external and internal edges. Feynman integrals are often organized in families labeled by integers a_1 through a_n below, which consist of all integrals sharing the same diagram but differing in the powers of certain momentum-dependent factors in the denominator.

A general form [6] of a Feynman integral family is the following:

$$I\left(a_1,\cdots,a_n\right)=\int\frac{\prod_{l=1}^{L}d^{D}k_l}{\prod_{i=1}^{n}\left[D_i\left(k_1^{\mu},\cdots,k_L^{\mu}\right)\right]^{a_i}}. \tag{1}$$

Since total derivatives integrate to zero, for any vector v^{μ} built from external momenta [23]:

$$0=\int\left(\prod_{\ell=1}^{L}d^{D}k_{\ell}\right)\frac{\partial}{\partial k_j^{\mu}}\left[\frac{v^{\mu}}{\prod_{i=1}^{n}\left[D_i\left(k_1^{\mu},\cdots,k_L^{\mu}\right)\right]^{a_i}}\right] \tag{2}$$

Expanding the derivative produces a linear relation among integrals in the same family [24]. Then, the Laporta algorithm [25] is used to order the integrals and solve the linear system. This is done iteratively, resulting in a set of integrals that cannot be further reduced by IBP. This set of integrals consists of what are called the master integrals. The original family of integrals from (1) is now:

$$I\left(\vec{a}\right)=\sum_{m}c_m\left(\{s,t,\cdots\},D\right)I_m^{(\text{master})} \tag{3}$$

with rational coefficient functions, c_m.

The master integrals in (3) serve as the fundamental building blocks of Feynman integrals [26]. Traditional reduction codes that can derive IBP identities exist [23] [25] [27], which use heuristics to select sets of optimal starting integrals, or seeds. Efficient seeding is crucial because it determines how completely and rapidly the system of IBP equations can be reduced to a set of master integrals, which would then be used to reconstruct the full set of Feynman integrals appearing in a given amplitude. However, existing programs can be computationally expensive, often requiring hundreds of thousands of CPU hours [6]. At the same time, advances in AI, driven by the rapid rise of large language models (LLMs) and chatbots, have shown a remarkable ability to generate text and code across diverse domains. This has motivated their exploration as tools to optimize IBP reduction by providing seeding strategies. For example, Funsearch [6] [28] enhances a genetic algorithm with a pre-trained LLM to propose novel seeding strategies. The inputs to the algorithm are two parent functions (v_0 and v_1), which are seeding strategies that already exist in the population, along with an incomplete function

v_2. The LLM completes v_2 and outputs a child program that proposes a new seeding strategy. Each resulting program represents a rule for deciding which integrals to include as seeds. Each candidate is then evaluated by an IBP solver, which assigns a fitness score based on whether the resulting system can be reduced and how many seeds are required. In a recent proof-of-concept test on a two-loop triangle-box example, FunSearch reduced the seed count from an initially unoptimized 14,588 to 476 after about 34 hours of runtime [6]. However, subsequent runs plateaued and produced irregular, partially redundant rules, indicating that without more structured guidance, the LLM tends to overfit to superficial patterns. The authors interpret this not as a definitive speed-up but as evidence that LLM-driven search can suggest new heuristics, such as constraints on the total index sum or counts of ones and zeros, that traditional genetic programming can then refine more efficiently. In this sense, FunSearch illustrates a promising direction for integrating generative AI into symbolic reduction pipelines, though current implementations remain exploratory and limited in reproducibility.

It is important to note that while LLM-guided approaches, such as FunSearch, demonstrate creative potential, their practical reliability remains limited. Outcomes depend heavily on prompt structure, random sampling, and the model pretraining corpus, making results difficult to reproduce and sensitive to implicit biases in training data. Current implementations therefore serve more as exploratory tools for generating candidate heuristics than as consistent, verifiable optimizers [6] [29].

After IBP reduction, our master integrals are functions of some kinematic variable such as x. We differentiate w.r.t x and use IBP again to express those derivatives as a linear combination of the master integrals.

$$\partial_x \mathbf{I}(x) = A(x)\mathbf{I}(x) \tag{4}$$

Then, we can change basis and put the system into a simpler form [30]:

$$\partial_x \mathbf{J}(x,\epsilon) = \epsilon \sum_i M_i \partial_x \log \phi_i(x) \mathbf{J}(x,\epsilon) \tag{5}$$

Therefore, the next step in Feynman integral evaluation is solving this system of first-order DEs. Analytical solutions are often impractical, so a more feasible approach is to evaluate these integrals numerically. However, traditional numerical methods, such as the widely used Monte Carlo integration with sector decomposition [31], can be computationally slow. These limitations have prompted the exploration of ML techniques. Recent work [32] has applied a framework based on the PIDL approach to approximate DEs numerically, where DNNs are used as universal function approximators to estimate solutions. An advantage of this approach is its flexibility, as it does not require expressing DEs in their canonical form (5) like analytical methods, and it also yields instantaneous evaluation times. The PIDL approach uses a pair of fully connected feedforward neural networks, one approximating the real parts and the other the imaginary parts of the master integral coefficients. The inputs to the networks are the kinematic variables, which

are invariants built from particle momenta and masses. The outputs are the predicted real and imaginary parts of the master integral coefficients, which can be used to reconstruct the full values of the Feynman integrals. The algorithm is trained with a dataset composed of two subsets: a boundary dataset, which comprises known master integral values at certain kinematic reference points, and a DE dataset, which is generated dynamically at each training loop. Applied to several one- and two-loop families, the approach achieves mean relative errors of roughly one percent, with training times on the order of an hour on a laptop-class GPU. Once trained, the model enables essentially instantaneous evaluations, suggesting clear efficiency advantages. However, Calisto *et al.* emphasize that accuracy control remains limited compared to analytical or sector-decomposition methods, making this a proof-of-concept demonstration rather than a production-ready computational tool [32]. Although the PIDL framework operates at the differential-equation level, both it and sector-decomposition approaches ultimately aim to numerically evaluate the same Feynman integrals. This proof-of-concept efficiency contrasts with modern sector-decomposition Monte Carlo frameworks such as SecDec [33]. Though the Monte Carlo framework routinely achieves sub-percent precision, exceeding the accuracy of the PIDL approach, it comes at the cost of substantially greater computational time through extensive stochastic sampling. The computational time often ranges from minutes to days, depending on diagram complexity. Therefore, while PIDL offers rapid inference once trained, Monte Carlo methods remain the benchmark for numerical accuracy and explicit uncertainty estimation.

In summary, Feynman integrals form the computational backbone of perturbative quantum field theory. Traditionally, generating seeds for IBP reduction has required extensive CPU resources, and DE evaluation has relied on canonical functions that are not always available. In recent years, advances in ML have provided new opportunities to address these limitations. On the IBP side, LLM-based methods have been explored for generating more efficient seeding strategies, showing potential to reduce seed counts but still facing challenges related to reproducibility, prompt dependence, and interpretability. On the DE side, the PIDL approach uses neural networks trained on boundary conditions and dynamic samples to evaluate master integrals, achieving near-instantaneous evaluation. Collectively, these advances illustrate how ML-based approaches can offer proof-of-concept pathways toward more scalable and efficient frameworks for Feynman integral evaluation in high-precision quantum field theory.

4. ML for Charged Particle Tracking

Charged particle tracking is the process of reconstructing the trajectories of charged particles, such as electrons, protons, pions, and muons, that are produced in collisions as they pass through a detector. As one of the most extensively studied tasks at the LHC, charged particle tracking is essential for many applications [34]. At the detector level, it provides basic observables such as track momenta and

charged particle multiplicities. Tracking is also critical for physics analyses, for example in identifying secondary vertices (used in searching for new particles) and in measuring particle lifetimes. The inner layers of the LHC detectors have magnetic fields that cause charged particles to bend and move in helical trajectories [4], allowing a precise determination of their momentum and charge. As the particles traverse the tracker layers, they leave localized hits, or detectable signals left in the detector material, through ionization energy deposits. As shown in **Figure 2**, track reconstruction involves systematically connecting these hits to form complete particle trajectories, effectively connecting the dots [35] to recover the paths of individual particles from measured detector signals.

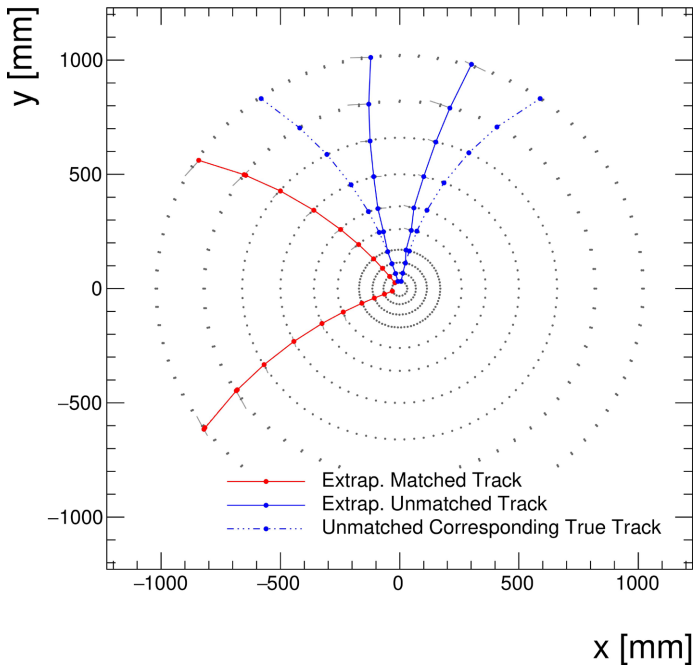

Figure 2. Charged particle tracking is illustrated: hits in the detector (dots) are systematically connected to reconstruct particle trajectories. Reproduced from Fig. 5 of [36].

Charged particle tracking in experiments is particularly challenging due to the large number of trajectories that must be reconstructed simultaneously. Detector hits do not carry an intrinsic identifier of the particle that created them, making the correct association of hits with their trajectories difficult [37]. Over the past few decades, experiments have relied on traditional computational techniques that use geometric pattern recognition and statistical track fitting to reconstruct tracks from raw hits. One of the most prominent methods used at the LHC is the Combinatorial Kalman Filter (CKF) [38], which combines track finding and track fitting in a search-tree-based algorithm. The CKF begins with short track seeds, typically formed from a few high-precision hits in the innermost detector layers. These seeds provide an estimate of the trajectory of the particle. From each seed, the candidate tracks are propagated outward through the detector, incorporating measurements from successive layers one at a time in a recursive prediction and

update cycle [39]. The process is repeated until the outermost layer is reached or a stopping condition is satisfied.

While the CKF remains the state-of-the-art algorithm for charged particle tracking in high-energy physics, it faces growing computational challenges under high-luminosity conditions. Its sequential, combinatorial nature leads to significant CPU time consumption as event complexity and pile-up increase [40]. To explore more scalable solutions, recent research has investigated machine-learning-based alternatives. In particular, charged particle tracking can be formulated as a graph-learning problem, where detector hits are represented as nodes and possible connections between them as edges [35]. Graph Neural Networks (GNNs) are well suited to this structure, as they can efficiently model the spatial relationships among hits to infer likely particle trajectories [41].

The Interaction Network (IN), a physics-motivated GNN architecture [42], has been applied to charged particle tracking by modeling all particles in a system simultaneously. Edge-Classifying Interaction Networks (EC-INs) [35] are an extension of the standard IN architecture specifically adapted for charged particle tracking. In this approach, the input is a network of hit "objects" as nodes and candidate connections between hits as edges. The relational and object reasoning steps of the IN are repurposed for edge and node re-embedding. The network infers the probability that each edge corresponds to a true track segment. A track building algorithm then leverages edge weights to form the full track candidates. The network is trained in a supervised fashion on simulated proton-proton collision events. During inference, edges exceeding a threshold are treated as true track segments, and clustering algorithms group hits into reconstructed track candidates. These candidates are then matched with simulated particles to extract track parameters such as transverse momentum and pseudorapidity. EC-IN and CKF operate in fundamentally different regimes; however, their comparison highlights a key architectural contrast. CKF relies on sequential, CPU-bound processing [43], whereas EC-IN leverages parallel graph-based execution on GPUs. In benchmark tests on simplified datasets such as TrackML, EC-IN achieves millisecond-scale inference with order-of-magnitude faster throughput than its CPU counterpart [35], illustrating the scalability potential of ML-based tracking approaches for future high-luminosity conditions. While the CKF pipeline remains the state of the art for reconstruction purity and stability, the EC-IN demonstrates how machine-learning-based, graph-oriented approaches can achieve competitive efficiency with significantly reduced inference latency, pointing to their potential for scalable, real-time tracking at the HL-LHC.

EuclidNet [7], a novel symmetry-equivariant GNN for charged particle tracking, builds on the graph-based formulation established by EC-INs by explicitly encoding the detector's cylindrical geometry. Using the graph representation of collision events and enforcing rotational symmetry around the detector beamline axis, EuclidNet [7] incorporates explicit SO(2) rotational symmetry, enabling parameter-efficient learning that maintains competitive accuracy with non-equivar-

iant approaches. EuclidNet is built by stacking multiple Euclidean Equivariant Blocks (EEBs) [44], which are units that ensure that the network's predictions remain consistent under geometric transformations. The inputs are 3D hit positions, with each hit split into two parts: the distance along the beamline (z-axis), which does not change under rotation, and the coordinates in the plane around the beamline, which rotate together with the event. Within these blocks, information is passed iteratively between nodes, using geometric relationships such as distances and dot products to guide learning while preserving rotational symmetry around the detector beamline. The network outputs a truth score for each edge, which represents the probability that the candidate connection corresponds to a real particle track [7]. EuclidNet is trained in a supervised manner using simulated proton-proton collision events with 3D hit positions and ground truth track information. During training, the network uses the binary cross-entropy loss to compare its predictions to the true labels. EuclidNet [7] performs graph-based edge classification using fewer than one thousand trainable parameters on HL-LHC-like simulated graphs, avoiding the combinatorial seeding and propagation that dominate CKF runtime. Model performance is evaluated using the area under the receiver-operating-characteristic curve (AUC), a standard classification metric that quantifies how well the network distinguishes true edges from false ones. The model achieves a slightly higher edge-classification AUC (approximately 0.991) [7] at small model scales than comparable non-equivariant benchmarks, though this advantage diminishes for larger architectures where unconstrained networks perform better. This behavior likely reflects the approximate rotational symmetry of realistic detectors, where strictly enforcing symmetry may constrain model expressivity in the presence of non-equivariant features. Despite these limitations, EuclidNet illustrates how physics-informed learning architectures may offer scalable alternatives as event complexity continues to grow at future collider experiments.

In summary, charged particle tracking is a central task in HEP. Traditional approaches such as the CKF are computationally expensive and scale poorly in high-occupancy environments like those at the LHC. ML approaches reframe charged particle tracking as a graph problem, applying advanced GNN architectures to model spatial and relational patterns among detector hits. Research using EC-INs and the symmetry-equivariant EuclidNet [7] has demonstrated promising accuracy and computational efficiency, offering scalable solutions for high-luminosity environments at the LHC and beyond. Continued progress in these methods will be crucial in advancing our understanding of matter and the fundamental structure of the universe.

5. ML for Jet Tagging

Another key application of ML in experimental HEP is jet tagging, a central classification task at the LHC. Jets are collimated sprays of particles produced when unstable particles created in proton-proton collisions decay, typically originating

from quarks, gluons, or hadronic decays of heavy particles. Jet tagging is the process of identifying the jet's initiating particle. Since jets from different particles have distinct features, the initiating particle can be inferred from the properties of the reconstructed jet [45]. Specifically, jets can originate from particles such as b quarks, c quarks, light quarks (u, d, s), and gluons [46]. The process of jet reconstruction and tagging from proton-proton collisions is shown in **Figure 3**.

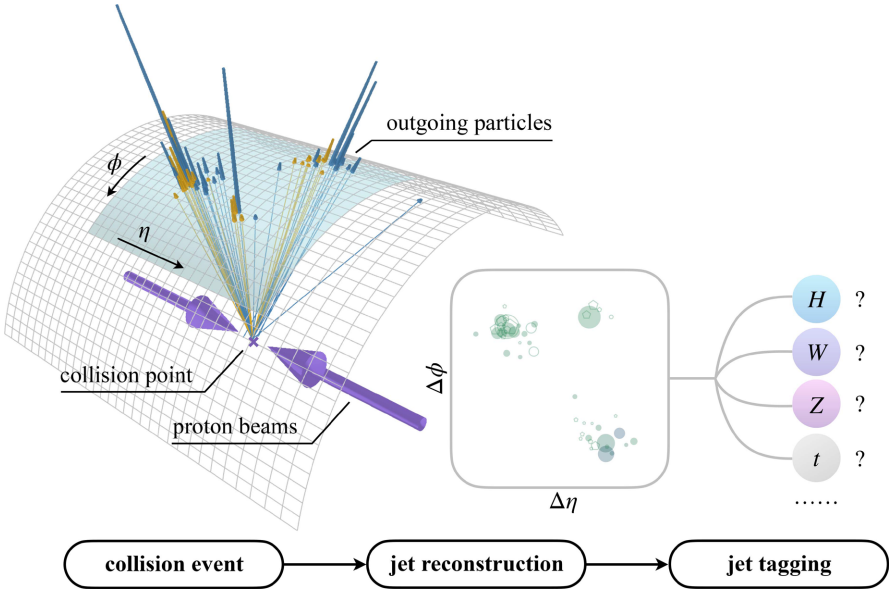

Figure 3. Jet tagging at the CERN LHC. Proton-proton collisions produce jets reconstructed by detector systems. Jet tagging classifies these jets and identifies their origin (e.g., Higgs, W/Z, or top). Reproduced from Fig. 1 from [47].

Jet tagging is a challenging task because partons produced in collisions radiate and fragment, generating jets composed of many outgoing particles. This additional radiation distorts the jet features, making it difficult to identify the initiating particle. Traditional jet tagging methods rely on observables inspired by Quantum Chromodynamics (QCD) [48]. These approaches typically use hand-crafted observables and simple multivariate classifiers, which make them sensitive to factors such as detector resolution and pileup. Although still used as baselines and for validation, they are increasingly being replaced by ML-based techniques [49]. A survey of how different ML architectures are applied to single-pronged jets (where the jet originates from a single parton or particle decay) is provided below.

Many neural network-based ML algorithms have been developed and tested for jet tagging. An example is the use of DNNs. In many DNN-based jet tagging algorithms [8], inputs include track-level measurements (e.g., momentum, direction, and displacement), secondary vertex properties, and other global jet kinematic features. The outputs are a set of probabilities that indicate the likelihood that a given jet belongs to one of several flavor categories, such as b, c, or light-flavor jets. Several well-developed DNN-based algorithms have been proposed for jet tagging, among which DeepCSV [50], a widely used CMS tagger; DL1 [51], the

first deep learning-based tagger by ATLAS; and DeepJet [52], a hybrid neural network in CMS, are among the most prominent examples. All three algorithms are multiclassifiers that are capable of distinguishing among several discrete categories as output rather than producing a single binary label. The algorithms are usually trained on simulated jets with known flavor labels and optimized using gradient-based methods to minimize loss [46]. DeepCSV and DL1 are limited because they use only a small number of charged tracks as input, which leads to a loss of information from neutral jet constituents and additional charged tracks. This limitation led to the development of the hybrid DeepJet algorithm, which integrates CNN, RNN, and DNN components. The convolutional filters process individual particle and vertex features. To capture sequential dependencies within the jet, the LSTM networks handle jets as sequences of constituents ordered by importance. The DNN then integrates these outputs along with global jet features to classify the jet flavor. In general, DNNs form the foundation of many modern jet taggers, integrating diverse jet features to learn complex correlations and achieve accurate classification.

Another class of neural networks applied in jet tagging is the CNN. Examples of well-developed CNN-based jet tagging algorithms include ImageTop [53], an image-based tagger, and DeepAK8 [53], a hybrid multiclassifier. Both have been deployed in the CMS experiment. ImageTop is based on a 2D CNN, which treats each jet as a two-dimensional image. It uses image recognition techniques to distinguish top quark jets from QCD background jets, which originate from generic quark and gluon fragmentation in strong-interaction processes. The inputs are jet images formed from energy deposits of charged/neutral hadrons, photons, electrons, and muons, along with additional kinematic variables. The output is the probability that the jet belongs to either the top quark or a QCD category. The network is trained on fully simulated CMS jets [54] using gradient-based optimization with a binary cross-entropy loss. The multiple convolutional layers effectively extract hierarchical and spatial features from jet images and enable reliable probability outputs. DeepAK8 extends the CNN-based approach to perform multiclass classification of hadronically decaying particles into five main categories: W, Z, H, t and other [53]. In this algorithm, each jet has two input lists. The first list, or particle list, consists of up to 100 jet constituent particles sorted by decreasing transverse momentum. The second list, or secondary vertex list, consists of up to seven secondary vertices, each described by kinematics, displacement, and quality criteria. DeepAK8 uses 1D CNNs to process these inputs, in contrast to the 2D image-based structure of ImageTop. The output of the convolutional layers is combined in a fully connected DNN, which produces the final probabilities [55] for classification. In general, CNNs enable jet taggers to exploit structured jet information effectively for efficient classification.

In addition to foundational neural network architectures such as DNNs, CNNs, and RNNs, many set- and graph-based ML approaches have been developed to better capture the structure of jet constituents. CNN- and RNN-based methods

represent jets as either images or ordered sequences of particles, often sorted by features such as transverse momentum. However, final-state particles in jets have no intrinsic order, so imposing one is arbitrary and may introduce biases. A more natural representation treats jets as unordered point clouds. An example of a set-based model is Deep Sets. Because these architectures are permutation-invariant, they are naturally well suited for modeling jets as unordered particle clouds [8]. An example of such a model is the Deep Impact Parameter Sets (DIPS) tagger [56], which was introduced in ATLAS after the LHC Run-2 data collection period. The inputs of this algorithm consist of track information represented as an unordered, variable-sized set. The output consists of three nodes, each representing the probability that the jet is a b, c, or light-flavor jet. DIPS is trained and evaluated on simulated top-antitop quark pair events produced in proton-proton collisions, and it is optimized by a gradient-based optimizer. Owing to its parallelizability and increased speed, DIPS is an excellent choice for identifying heavy-flavor jets and has the potential to improve computational efficiency in ATLAS reconstruction.

The ATLAS experiment has developed several graph-based approaches using GNNs. Two examples are GN1 and GN2. The GN1 [57] algorithm, introduced for LHC Run-3, uses information from tracks within a jet to predict its flavor. In contrast to earlier techniques that process jet information sequentially (RNNs) or in a flattened fashion (DNNs), this method is graph-based because it treats tracks as nodes in a fully connected graph and learns inter-track relationships through graph layers. The inputs to GN1 include the transverse momentum and pseudorapidity of the jet, as well as a variable number of tracks per jet. Each track is described by 21 features, such as kinematics, impact parameters with significance, and hit counts across tracker layers. GN1 is trained on 30 million simulated proton-proton collision jets, with resampled, normalized, and shuffled kinematic variables to ensure effective learning and evaluation. Building on the GN1 architecture, the ATLAS Collaboration introduced the GN2 algorithm [58], which replaces the graph attention network core with a Transformer-based encoder [59]. Transformers use attention mechanisms to capture dependencies among jet constituents more effectively within the graph-based framework. In GN2, the attention mechanism is enhanced with a dense layer between the attention layers, which increases flexibility and enables the model to learn more complex relationships among jet constituents. These enhancements make GN2 a more advanced graph-based flavor tagging algorithm. It performs better in simulated studies compared to GN1 [58], particularly for light-flavor jet rejection.

A key practical challenge in jet tagging is domain shift. It arises because taggers are trained on simulation but applied to collision data, where subtle differences in detector response, pile-up noise, and particle-shower modeling can degrade performance. Even state-of-the-art taggers therefore require experimental calibration, deriving scale factors from real events to correct simulation-based predictions and ensure closure with data distributions [60]. In addition, mismodeled

inputs can bias ML taggers. Mitigation strategies such as adversarial training and decorrelation from kinematic or mass variables have been explored to improve robustness under realistic conditions [61] [62]. Without such mitigations, the high-dimensional learned representations may amplify simulation biases, reducing confidence in data performance and inflating systematic uncertainties.

In summary, jet tagging remains a central task in HEP, as accurately identifying the initiating particle is essential for probing fundamental interactions and searching for new phenomena. Traditional taggers rely on hand-crafted observables and simple classifiers, whereas ML-based methods learn complex, high-dimensional features directly from jet constituents. These approaches tend to be more robust to detector effects and scale better with increasing event complexity. Modern ML taggers have demonstrated superior efficiency and background rejection compared to traditional QCD-inspired techniques.

6. Conclusions

The application of ML to HEP is influencing both theoretical and experimental workflows at the LHC by addressing challenges that traditional methods struggle to resolve efficiently. In theoretical calculations for Feynman integral evaluation, LLMs have begun to be explored as a method for generating more efficient seeding strategies for IBP reductions [6], while physics-informed neural networks can provide promising fast and accurate approximations to DE solutions for master integrals [32]. At the detector level, graph-based models such as EC-INs [35] and symmetry-equivariant architectures such as EuclidNet [7] have shown promising scalability in high-pileup/high-occupancy simulations for charged particle tracking. In classification tasks, jet tagging has seen marked advances through the application of deep, convolutional, recurrent, graph, and transformer-based networks, which collectively extract high-dimensional jet substructure information more effectively than many traditional hand-crafted observables.

ML in HEP faces constraints that differ from other domains, such as image recognition [63]or protein folding [64]. Collision data are produced at extreme rates, far beyond what can be permanently stored, and training often relies on simulations rather than large labeled datasets [65] [66]. These conditions limit the straightforward transfer of standard ML techniques, but underscore the need for approaches tailored to the structure and symmetries of physical systems. With the development of physics-informed architectures and efficient, low-latency models [10], there remains substantial opportunity for further progress. As a result, HEP continues to serve as both a uniquely challenging and fertile ground for innovation.

The advances highlighted in this review demonstrate both the immediate impact and the long-term potential of ML in HEP. Extending physics-informed and LLM-based methods to more complex Feynman integrals, optimizing graph-based tracking architectures for real-time deployment, and integrating set- and graph-based representations for jet tagging represent promising directions for future work. At the same time, improving interpretability and maintaining con-

sistency with established physical principles will remain central challenges as ML models grow in scale and complexity.

The impact of ML on HEP extends beyond computational efficiency. By making previously intractable calculations more accessible and facilitating faster analysis of increasingly complex collision datasets, ML is expanding the scope of experimentally and theoretically accessible questions. These approaches facilitate more precise tests of the Standard Model and enhance the sensitivity of searches for phenomena beyond it. Conversely, HEP serves as a rigorous testbed for the development of ML itself. The scale of available data, the requirement for interpretability, and the need to embed physical symmetries and conservation laws drive the creation of algorithms that are more robust, efficient, and generalizable. In this reciprocal relationship, ML empowers particle physics to probe the structure of matter with enhanced precision in selected applications, while the challenges of physics continue to stimulate innovations that advance the broader field of ML.

While this review has focused primarily on Feynman integral evaluation, charged particle tracking, and jet tagging, these topics represent just one subset of ML applications in HEP among a broader ecosystem. Other active areas of research [10], including fast detector simulation, anomaly detection, and theory-assisted discovery, illustrate the breadth and diversity of ongoing efforts. Much remains to be explored, and future developments will undoubtedly reveal new ways in which ML can advance both the frontiers of physics and the capabilities of computation.

Acknowledgements

The authors thank Professor Phiala E. Shanahan of the Department of Physics at the Massachusetts Institute of Technology for her insightful guidance and the many fruitful discussions that shaped the development of this review. Her mentorship was invaluable in strengthening our understanding of both the underlying physics and the applications of machine learning in this field.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] CERN (2025) The Large Hadron Collider. https://home.cern/science/accelerators/large-hadron-collider

[2] Aad, G., *et al.* (2012) Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC. *Physics Letters B*, **716**, 1-29.

[3] Ohl, T. (1999) Vegas Revisited: Adaptive Monte Carlo Integration beyond Factorization. *Computer Physics Communications*, **120**, 13-19. https://doi.org/10.1016/s0010-4655(99)00209-x

[4] Heintz, A., *et al.* (2020) Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs. 34*th Conference on Neural Information Processing Systems*,

Vancouver, December 2020.

[5] de Oliveira, L., Kagan, M., Mackey, L., Nachman, B. and Schwartzman, A. (2016) Jet-images—Deep Learning Edition. *Journal of High Energy Physics*, **2016**, Article No. 69. https://doi.org/10.1007/jhep07(2016)069

[6] von Hippel, M. and Wilhelm, M. (2025) Refining Integration-by-Parts Reduction of Feynman Integrals with Machine Learning. *Journal of High Energy Physics*, **2025**, Article No. 185. https://doi.org/10.1007/jhep05(2025)185

[7] Murnane, D., Thais, S. and Thete, A. (2023) Equivariant Graph Neural Networks for Charged Particle Tracking. 21*st International Workshop on Advanced Computing and Analysis Techniques in Physics Research*: *AI meets Reality*, Bari, October 2022.

[8] Mondal, S. and Mastrolorenzo, L. (2024) Machine Learning in High Energy Physics: A Review of Heavy-Flavor Jet Tagging at the LHC. *The European Physical Journal Special Topics*, **233**, 2657-2686. https://doi.org/10.1140/epjs/s11734-024-01234-y

[9] Turing, A.M. (1950) Computing Machinery and Intelligence. *Mind*, **49**, 433-460. https://courses.cs.umbc.edu/471/papers/turing.pdf

[10] Grosso, G., Harris, P., Mishra-Sharma, S. and Shanahan, P. (2024) A Virtuous Cycle: Generative AI and Discovery in the Physical Sciences. An MIT Exploration of Generative AI. https://mit-genai.pubpub.org/pub/ewp5ckmf

[11] Picton, P. (1994) What Is a Neural Network? In: Picton, P., Ed., *Introduction to Neural Networks*, Macmillan Education, 1-12. https://doi.org/10.1007/978-1-349-13530-1_1

[12] Oluleye, B.I., Chan, D.W.M. and Antwi-Afari, P. (2023) Adopting Artificial Intelligence for Enhancing the Implementation of Systemic Circularity in the Construction Industry: A Critical Review. *Sustainable Production and Consumption*, **35**, 509-524. https://doi.org/10.1016/j.spc.2022.12.002

[13] O'Shea, K. and Nash, R. (2015) An Introduction to Convolutional Neural Networks. arXiv:1511.08458.

[14] Schmidt, R.M. (2019) Recurrent Neural Networks (RNNs): A Gentle Introduction and Overview. arXiv:1912.05911.

[15] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[16] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., *et al.* (2021) Graph Neural Networks: A Review of Methods and Applications. *AI Open*, **1**, 57-81. https://doi.org/10.1016/j.aiopen.2021.01.001

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2023) Attention Is All You Need. arXiv:1706.03762.

[18] Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2017) Physics Informed Deep Learning (Part I): Data-Driven Solutions of Nonlinear Partial Differential Equations. arXiv:1711.10561.

[19] Glashow, S.L. (1961) Partial-Symmetries of Weak Interactions. *Nuclear Physics*, **22**, 579-588. https://doi.org/10.1016/0029-5582(61)90469-2

[20] Weinberg, S. (1967) A Model of Leptons. *Physical Review Letters*, **19**, 1264-1266. https://doi.org/10.1103/physrevlett.19.1264

[21] Salam, A. (1968) Weak and Electromagnetic Interactions. *Conference Proceedings C*, **680519**, 367-377.

[22] Henn, J.M. (2015) Lectures on Differential Equations for Feynman Integrals. *Journal of Physics A*: *Mathematical and Theoretical*, **48**, Article 153001.

https://doi.org/10.1088/1751-8113/48/15/153001

[23] Chetyrkin, K.G. and Tkachov, F.V. (1981) Integration by Parts: The Algorithm to Calculate β-Functions in 4 Loops. *Nuclear Physics B*, **192**, 159-204. https://doi.org/10.1016/0550-3213(81)90199-1

[24] Grozin, A.G. (2011) Integration by Parts: An Introduction. *International Journal of Modern Physics A*, **26**, 2807-2854. https://doi.org/10.1142/s0217751x11053687

[25] Laporta, S. (2000) High-Precision Calculation of Multiloop Feynman Integrals by Difference Equations. *International Journal of Modern Physics A*, **15**, 5087-5159. https://doi.org/10.1142/s0217751x00002159

[26] Barrera, O., Dersy, A., Husain, R., *et al.* (2025) Analytic Regression of Feynman Integrals from High-Precision Numerical Sampling. arXiv:2507.17815.

[27] Tkachov, F.V. (1981) A Theorem on Analytical Calculability of 4-Loop Renormalization Group Functions. *Physics Letters B*, **100**, 65-68. https://doi.org/10.1016/0370-2693(81)90288-4

[28] Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M.P., Dupont, E., *et al.* (2024) Mathematical Discoveries from Program Search with Large Language Models. *Nature*, **625**, 468-475. https://doi.org/10.1038/s41586-023-06924-6

[29] Gundersen, O.E., Coakley, K., Kirkpatrick, C. and Gil, Y. (2023) Sources of Irreproducibility in Machine Learning: A Review. arXiv:2204.07610.

[30] Gehrmann, T. and Remiddi, E. (2000) Differential Equations for Two-Loop Four-Point Functions. *Nuclear Physics B*, **580**, 485-518. https://doi.org/10.1016/s0550-3213(00)00223-6

[31] Binoth, T. and Heinrich, G. (2000) An Automatized Algorithm to Compute Infrared Divergent Multi-Loop Integrals. *Nuclear Physics B*, **585**, 741-759. https://doi.org/10.1016/s0550-3213(00)00429-6

[32] Calisto, F., Moodie, R. and Zoia, S. (2024) Learning Feynman Integrals from Differential Equations with Neural Networks. *Journal of High Energy Physics*, **2024**, Article No. 124. https://doi.org/10.1007/jhep07(2024)124

[33] Carter, J. and Heinrich, G. (2011) Secdec: A General Program for Sector Decomposition. *Computer Physics Communications*, **182**, 1566-1581. https://doi.org/10.1016/j.cpc.2011.03.026

[34] Cassel, D.G., *et al.* (1986) Report of the Central Tracking Group. *Conference Proceedings C*, **860623**, 377.

[35] DeZoort, G., Thais, S., Duarte, J., Razavimaleki, V., Atkinson, M., Ojalvo, I., *et al.* (2021) Charged Particle Tracking via Edge-Classifying Interaction Networks. *Computing and Software for Big Science*, **5**, Article No. 26. https://doi.org/10.1007/s41781-021-00073-z

[36] Abidi, H., Boveia, A., Cavaliere, V., Furletov, D., Gekow, A., Kalderon, C.W. and Yoo, S. (2022) Charged Particle Tracking with Machine Learning on FPGAs. arXiv:2212.02348.

[37] Amrouche, S., Kiehn, M., Golling, T. and Salzburger, A. (2021) Hashing and Metric Learning for Charged Particle Tracking. 33*rd Annual Conference on Neural Information Processing Systems*, Vancouver, December 2019.

[38] Billoir, P. (1989) Progressive Track Recognition with a Kalman-Like Fitting Procedure. *Computer Physics Communications*, **57**, 390-394. https://doi.org/10.1016/0010-4655(89)90249-x

[39] Frühwirth, R. and Strandlie, A. (2006) Application of Adaptive Filters to Track Finding. *Nuclear Instruments and Methods in Physics Research Section A*: *Accelerators*,

Spectrometers, Detectors and Associated Equipment, **559**, 162-166. https://doi.org/10.1016/j.nima.2005.11.135

[40] Heinrich, L., Huth, B., Salzburger, A. and Wettig, T. (2024) Combined Track Finding with GNN & CKF. 8*th International Connecting the Dots Workshop* (CTD 2023), Toulouse, October 2023.

[41] Shlomi, J., Battaglia, P. and Vlimant, J.-R. (2020) Graph Neural Networks in Particle Physics. *Machine Learning: Science and Technology*, **2**, Article 021001.

[42] Battaglia, P.W., Pascanu, R., Lai, M., Rezende, D. and Kavukcuoglu, K. (2016) Interaction Networks for Learning about Objects, Relations and Physics. arXiv:1612.00222.

[43] Mlinarevic, M. (2024) The ATLAS Inner Detector Trigger Performance in Run 3. *Proceedings of* 12*th Large Hadron Collider Physics Conference—PoS*(*LHCP*2024), Boston, 3-7 June 2024, 1-6. https://doi.org/10.22323/1.478.0220

[44] Hordan, S., Amir, T. and Dym, N. (2024) Weisfeiler Leman for Euclidean Equivariant Machine Learning. arXiv:2402.02484.

[45] Qu, H. and Gouskos, L. (2020) Jet tagging via particle clouds. *Physical Review D*, **101**, Article 056019. https://doi.org/10.1103/physrevd.101.056019

[46] Stoye, M. (2018) Deep Learning in Jet Reconstruction at CMS. *Journal of Physics: Conference Series*, **1085**, Article 042029. https://doi.org/10.1088/1742-6596/1085/4/042029

[47] Qu, H., Li, C. and Qian, S. (2022) Particle Transformer for Jet Tagging. arXiv:2202.03772.

[48] Usman, M., Shahid, M.H., Ejaz, M., Hani, U., Fatima, N., Khan, A.R., Khan, A. and Mirza, N.M. (2024) Particle Multi-Axis Transformer for Jet Tagging. arXiv:2406.06638.

[49] Wu, Y., Wang, K., Li, C., Qu, H. and Zhu, J. (2025) Jet Tagging with More-Interaction Particle Transformer. *Chinese Physics C*, **49**, Article 013110. https://doi.org/10.1088/1674-1137/ad7f3d

[50] Sirunyan, A.M., *et al.* (2018) Identification of Heavy-Flavour Jets with the CMS Detector in pp Collisions at 13 TeV. *Journal of Instrumentation*, **13**, P05011.

[51] CERN (2017) Optimisation and Performance Studies of the ATLAS *b*-Tagging Algorithms for the 2017-18 LHC Run. Technical Report. https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-013

[52] Bols, E., Kieseler, J., Verzetti, M., Stoye, M. and Stakia, A. (2020) Jet Flavour Classification Using Deepjet. *Journal of Instrumentation*, **15**, P12012-P12012. https://doi.org/10.1088/1748-0221/15/12/p12012

[53] Sirunyan, A.M., *et al.* (2020) Identification of Heavy, Energetic, Hadronically Decaying Particles Using Machine-Learning Techniques. *Journal of Instrumentation*, **15**, P06005.

[54] Kagan, M. (2020) Image-Based Jet Analysis. arXiv:2012.09719.

[55] Cagnotta, A., Carnevali, F. and De Iorio, A. (2022) Machine Learning Applications for Jet Tagging in the CMS Experiment. *Applied Sciences*, **12**, Article 10574. https://doi.org/10.3390/app122010574

[56] CERN (2020) Deep Sets Based Neural Networks for Impact Parameter Flavour Tagging in ATLAS. Technical Report. https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2020-014

[57] (2022) Graph Neural Network Jet Flavour Tagging with the ATLAS Detector.

[58] Aad, G., *et al.* (2025) Transforming Jet Flavour Tagging at ATLAS. arXiv:2505.19689.

[59] Han, K., Xiao, A., Wu, E., Guo, J., Xu, C. and Wang, Y. (2021) Transformer in Transformer. arXiv:2103.00112.

[60] Aad, G., Abbott, B., *et al.* (2023) Calibration of the Light-Flavour Jet Mistagging Efficiency of the b-Tagging Algorithms with Z + Jets Events Using 139 fb^{-1} of Atlas Proton-Proton Collision Data at $\sqrt{s}=13$ TeV. *The European Physical Journal C*, **83**, Article No. 728.

[61] Stein, A., Coubez, X., Mondal, S., Novak, A. and Schmidt, A. (2022) Improving Robustness of Jet Tagging Algorithms with Adversarial Training. *Computing and Software for Big Science*, **6**, Article No. 15. https://doi.org/10.1007/s41781-022-00087-1

[62] Shimmin, C., Sadowski, P., Baldi, P., Weik, E., Whiteson, D., Goul, E., *et al.* (2017) Decorrelated Jet Substructure Tagging Using Adversarial Neural Networks. *Physical Review D*, **96**, Article 074034. https://doi.org/10.1103/physrevd.96.074034

[63] Deng, J., Dong, W., Socher, R., Li, L., Li, K. and Li, F.-F. (2009) ImageNet: A Large-Scale Hierarchical Image Database. 2009 *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, 20-25 June 2009, 248-255. https://doi.org/10.1109/cvpr.2009.5206848

[64] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., *et al.* (2021) Highly Accurate Protein Structure Prediction with Alphafold. *Nature*, **596**, 583-589. https://doi.org/10.1038/s41586-021-03819-2

[65] Guest, D., Cranmer, K. and Whiteson, D. (2018) Deep Learning and Its Application to LHC Physics. *Annual Review of Nuclear and Particle Science*, **68**, 161-181. https://doi.org/10.1146/annurev-nucl-101917-021019

[66] Radovic, A., Williams, M., Rousseau, D., Kagan, M., Bonacorsi, D., Himmel, A., *et al.* (2018) Machine Learning at the Energy and Intensity Frontiers of Particle Physics. *Nature*, **560**, 41-48. https://doi.org/10.1038/s41586-018-0361-2