Research Square

Preprints are preliminary reports that have not undergone peer review. They should not be considered conclusive, used to inform clinical practice, or referenced by the media as validated information.

Artificial intelligence for improved fitting of trajectories of elementary particles in dense materials immersed in a magnetic field

Saúl Alonso Monsalve (saul.alonso.monsalve@cern.ch)
ETH Zurich https://orcid.org/0000-0002-9678-7121
Davide Sgalaberna
ETH Zurich https://orcid.org/0000-0001-6205-5013
Xingyu Zhao
ETH Zurich
Clark McGrew
Stony Brook University
André Rubbia
ETH Zurich

Article

Keywords:

Posted Date: January 23rd, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2326080/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

Version of Record: A version of this preprint was published at Communications Physics on May 26th, 2023. See the published version at https://doi.org/10.1038/s42005-023-01239-4.

Artificial intelligence for improved fitting of trajectories of elementary particles in dense materials immersed in a magnetic field

Saúl Alonso-Monsalve,[1,*] Davide Sgalaberna,[1] Xingyu Zhao,[1] Clark McGrew,[2] and André Rubbia[1]

[1]*ETH Zurich, Institute for Particle physics and Astrophysics, CH-8093 Zurich, Switzerland.*

[2]*State University of New York at Stony Brook, Department of Physics and Astronomy, Stony Brook, New York, USA.*

In this article, we use artificial intelligence algorithms to show how to enhance the resolution of the elementary particle track fitting in dense detectors, such as plastic scintillators. We use deep learning to replace more traditional Bayesian filtering methods, drastically improving the reconstruction of the interacting particle kinematics. We show that a specific form of neural network, inherited from the field of natural language processing, is very close to the concept of a Bayesian filter that adopts a hyper-informative prior. Such a paradigm change can influence the design of future particle physics experiments and their data exploitation.

I. INTRODUCTION

Understanding the behaviour of subatomic particles traversing dense materials, often immersed in magnetic fields, has been crucial to their discovery, detection, identification and reconstruction, and it is a critical component for exploiting any particle detector [1–6]. Modern radiation detectors have evolved towards "imaging detectors", in which elementary particles leave individual traces called "tracks" [7–11]. These imaging detectors require a "particle flow" reconstruction: particle signatures are precisely reconstructed in three dimensions, and the kinematics (energy and momentum vector) of the primary particle can be measured track-by-track. It also means that a more significant amount of details can be obtained on each particle. These features open the question of which methods are best suited to handle the "images" created by the subatomic particles.

Common Monte Carlo (MC) based methods used in the track fitting flow belong to the family of Bayesian filters and, more specifically, they are extensions to the standard Kalman filter [12] or particle filters algorithms, with special mention to the Sequential Importance Resampling particle filter (SIR-PF) [13]. The knowledge about how an electrically charged subatomic particle propagates through a medium (i.e., the energy loss, the effect of multiple scattering, and the curvature due to magnetic field) can be embedded into a prior (often in the form of a covariance matrix for Kalman filters). In particle filters, the nodes of the track are fitted sequentially: given a node state, the following node in the particle track is obtained by throwing random samples - known as "particles" - and making a guess of the following state by applying a likelihood between the sampled particles and the data (which could be, for instance, the signatures obtained from the detector readout channels). The result can be the position of the fitted nodes of a particle track or directly its momentum vector and its electric charge. Usually, the problem is simplified using a prior that follows a Gaussian distribution, like in the Kalman filter, which also considers a simplified version of the detector geometry. Examples can be found in [14–16]. However, the filtering is not trivial since both the particle energy loss and multiple scattering angles depend on the momentum, which changes fast in dense materials, and approximations are often necessary. Moreover, it is hard to incorporate finer details of a realistic detector geometry and response (e.g., signal crosstalk between channels, air gaps in the detector active volume, presence of different materials, or non-uniformities in the detector response as a function of the particle position, inhomogeneous magnetic field) or to deal with deviations in the particle trajectory due to the emission of high-energy δ-rays, with photon Bremsstrahlung emission, with the Bragg peak of a stopping particle, or with inelastic interactions. All these pieces of information are available in the simulation of a particle physics experiment [17–21] and can be validated or tuned with data but it is not straight-forward to use them in the reconstruction of the particle interaction. Hence, developing new reconstruction methods capable of analysing all the information available becomes essential.

The most promising solution is given by artificial intelligence and, more specifically, by deep learning, a subfield of machine learning based on artificial neural networks [22–25]. Initially inspired by how the human brain functions, these mathematical algorithms can efficiently extract complex features in a multi-dimensional space after appropriate training. Neural networks (NNs) have been found to be particularly successful in the reconstruction and analysis of particle physics experiments [26–30]. Thus far, deep learning has been used in high-energy physics (HEP) for tasks such as classification [27, 31–33], semantic segmentation [30, 34], or regression [35–37]. Typically, the raw detector signal is analysed to extract the physics information. This approach is quite common in experiments studying neutrinos, for example, to classify the flavour of the interaction (ν_μ, ν_e, or ν_τ) by using convolutional neural networks (CNNs) [27, 31, 38], or the different types of signatures observed in the detec-

* E-mail: saul.alonso.monsalve@cern.ch

tor [30, 34]. These methods have been shown to outperform more traditional ones, such as likelihood inference or decision trees. However, asking a neural network to extract high-level physics information directly from the raw signatures left in the detector by the charged particles produced by a neutrino interaction is conceivable as challenging. An example is the neutrino flavour identification (as mentioned before), which incorporates diverse contributions, from the modelling of the neutrino interaction cross-section to the propagation of the particles in matter and, finally, the particular response of the detector. Expecting a neural network to learn and parametrise all these contributions could become unrealistic and lead to potential deficiencies.

An alternative and promising approach is to use deep learning to assist the more traditional particle flow methods in reconstructing particle propagation, which consists of a chain of different analysis steps that can include the three-dimensional matching of the voxelised signatures in the detector readout 2D views, the definition of more complex objects such as tracks and, finally, the fit of the track in order to reconstruct the particle kinematics. As described above, the last step is critical and is usually performed by a Bayesian filter that has to contain as much information as possible in its multi-dimensional prior. It becomes clear that, overall, the reconstruction performance depends on the detector design (e.g., granularity or detection efficiency) and on the a priori knowledge of the particle propagation in the detector, the prior. Although prohibitive for traditional Bayesian filters, the problem of parameterising a high-dimensional space can be overcome with deep learning since neural networks can be explicitly designed for it.

Even though the generic idea of using deep learning as an alternative to Bayesian filtering has already been explored [39], common applications focus on tasks such as enhancing and predicting vehicle trajectories [40, 41]. Furthermore, the closest application we can currently find in HEP and other fields like biology is to use deep learning to perform "particle tracking" [42–44], which relies on connecting detected hits to form and select particles, distinct from the idea of fitting the detected hits to obtain a good approximation to the actual particle trajectory.

In this article, we propose the design of a recurrent neural network (RNN) and a Transformer to fit particle trajectories. We found that these neural nets, inherited from the field of natural language processing, are very close to the concept of a Bayesian filter that adopts a hyper-informative prior. Hence, they become excellent tools for drastically improving the accuracy and resolution of elementary particle trajectories.

II. PROOF-OF-PRINCIPLE

In order to train and test the developed neural networks and compare their performance with a more classical Bayesian filter, an idealized three-dimensional fine-grain plastic scintillator detector was taken as a case study. We simulated a cubic detector composed of a homogeneous plastic scintillator with a size of $2 \times 2 \times 2\ m^3$. A uniform magnetic field is applied, aligned to one axis of the detector (X-axis) and its strength is chosen to be 0.5 T. The detector is divided into small cubes of size 1 cm^3, summing $200 \times 200 \times 200$ cubes in total. Each cube is assumed to be equipped with a sensor that collects the scintillation light produced when a particle traverses it. We simulate the signals read from each sensor and reconstruct the event based on these signals. The track input to the fitters will be extracted from event reconstruction.

Overall, the simulation and reconstruction are divided into three steps:

1. **Energy deposition simulation**: this step uses the Geant4 toolkit [17–19] to simulate particle trajectories in the detector and their energy deposition along the path.

2. **Detector response simulation**: this step simulates detector effects and converts the energy deposition into signals the detector can receive. The current detector effect being considered is the light leakage from one cube to the adjacent one (named crosstalk). The leakage probability per face is assumed to be 3%. The energy deposition is converted from the physics unit (MeV) into the "signal unit" (depending on the detector) by using a constant factor, which is fixed to be 100 / MeV for this analysis. Besides, a threshold is also implemented on the sensor, requiring that at least one signal unit be received to activate the sensor.

3. **Reconstruction**: this step takes the signals generated from the former steps and reconstructs objects, such as tracks, that can be input to the fitter. Starting from 3D "cube hits" (what we have after the detector response simulation), we then apply the following two methods to find track segments from the whole event: (1) the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [45], which groups hits into large clusters that, in each cluster, all hits are adjacent to each other; (2) the minimum spanning tree (MST) [46] for each cluster to order hits and break the cluster into smaller track segments at each junction point. Afterwards, the primary track segment will be selected for track fitter input.

The simulation and reconstruction processes resulted in single-charged particles (protons, pions $\pi^{\pm}$, muons $\mu^{\pm}$, and electrons $e^{\pm}$) starting at random positions in the detector active volume with isotropic directions and uniform distributions of their initial momentum: between 0 and 1.5 GeV/c (protons), 0 and 1.5 GeV/c (pions), 0 and 2.5 GeV/c (muons) and 0 and 3.5 GeV/c (electrons). Each particle consisted of a number of reconstructed 3D hits belonging to the track, where each hit is represented

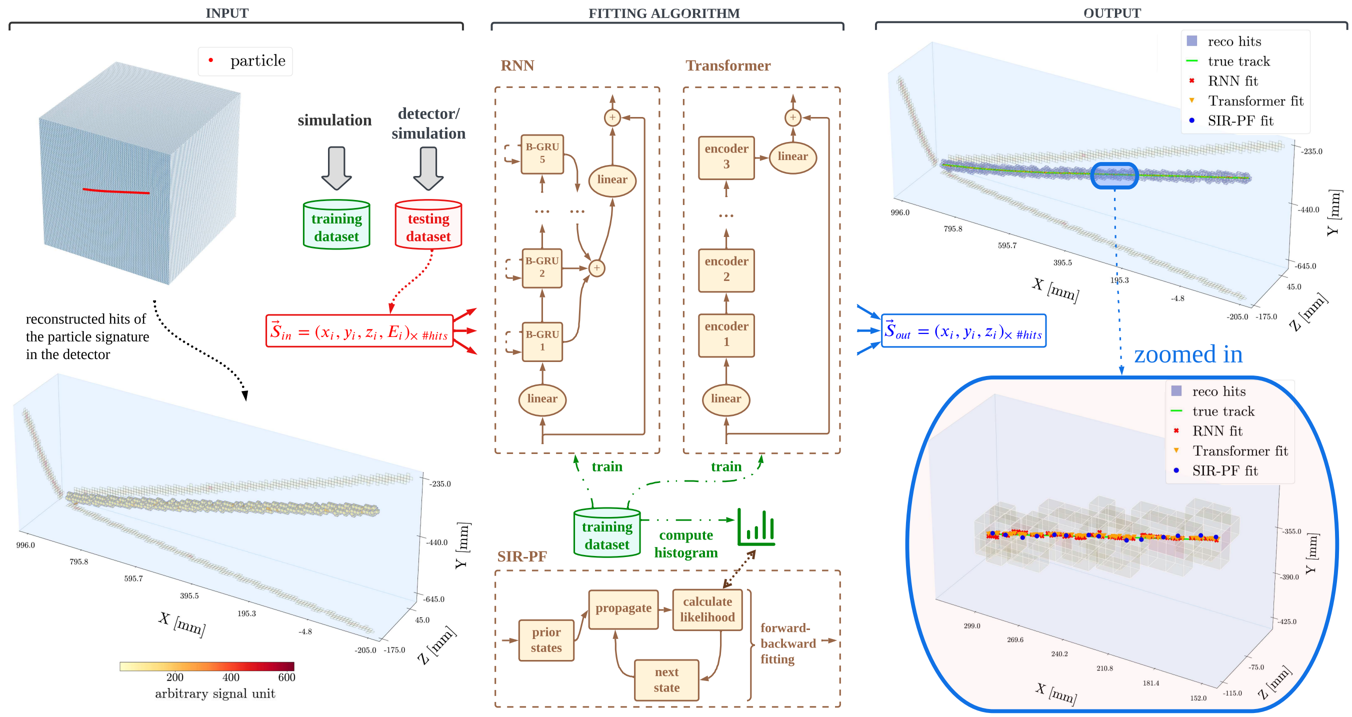

FIG. 1. Workflow of a crossing muon track fitting using the three algorithms: recurrent neural network (RNN), Transformer, and Sequential Importance Resampling particle filter (SIR-PF). From left to right, the diagram shows the steps from the particle simulation/detection until the particle is fitted using the different algorithms.

by a three-dimensional spatial position and an energy deposition in an arbitrary signal unit. For each reconstructed hit in a particle, there is a **true node** (to be learnt during the supervised training) which represents the closest 3D point to the hit in the actual particle trajectory; in that way, there is a 1-to-1 correspondence between reconstructed hits (even for crosstalk) and true nodes. We refer in the rest of the article to the output of the algorithms developed as **fitted nodes**, which form the fitted trajectory for each particle.

III. RESULTS

In this section, we discuss the performance of a recurrent neural network (RNN) [47–49] and a Transformer [50], comparing their results with the ones from a custom SIR-PF (as described in Sec. I). The developed methods, described in detail in Sec. V, were run on a test dataset of simulated elementary particles (statistically independent of the dataset used for training) consisting of 1,759,491 particles (412,092 protons, 432,807 pions $\pi^\pm$, 447,003 $\mu^\pm$, and 467,589 $e^\pm$). For each simulated particle, the goal was to use the reconstructed hits to predict the actual track trajectory and then to analyse its physics impact on the detector performance, as described later in this section. The output of the different methods was a list of fitted nodes, i.e. the predicted 3D positions of the elementary particle in the detector. A visual example of the particle trajectory fitting using the different techniques is shown in Fig. 1.

A. Fitting of the particle trajectory

For the SIR-PF, we have considered two different scenarios that vary in the reconstructed input information to the filter: (1) all the reconstructed 3D hits are used as input; (2) only real track hits[i] are used as input, which is unavailable information for actual data (and represents a nonphysical scenario) but allows us to test the ideal performance for the current filter. The input for the RNN and Transformer always consisted of all the reconstructed 3D hits. Figure 2 shows a comparison of the performance for the three methods (considering the SIR-PF variant with all the reconstructed hits as input). The results indicate that the Transformer outperforms the other techniques (even for the case with only track hits). Besides, the RNN reports significantly better results than the SIR-PF with only track hits used as input and slightly better fittings concerning the SIR-PF with all hits used as input, which demonstrates not only that the NN-based approaches can handle crosstalk hits but also go beyond

[i] With "real track hits" we refer to hits from cubes the actual particle has passed through.

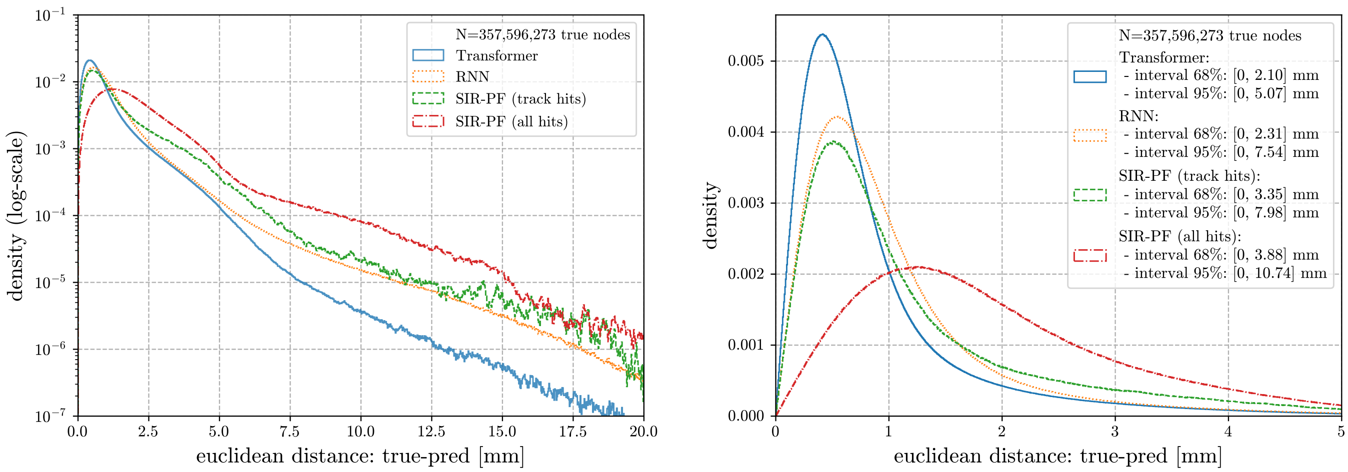

FIG. 2. The distribution of the three-dimensional Euclidean distance between the actual elementary particle position and the corresponding fitted node predicted by the Transformer, the recurrent neural network (RNN), and the Sequential Importance Resampling particle filter (SIR-PF, with only track hits and all hits as input). The sample used to generate the histograms contains all the simulated particles. Results show the distributions for a log-scale (left) and normal-scale (right, cropped to a maximum distance of 5 mm) densities, as well as the one-sided area ranges, representing 68% and 95% of the distributions.

algorithm	input	particle	mean (μ) [mm]		std (σ) [mm]		68% area [mm]		95% area [mm]	
Transformer	**all hits**	all	0.92		0.97		[0, 2.10]		[0, 5.07]	
			stopping	escaping	stopping	escaping	stopping	escaping	stopping	escaping
		“	1.10	0.80	1.16	0.78	[0, 2.57]	[0, 1.64]	[0, 5.76]	[0, 4.30]
		p	1.11	0.81	1.18	0.80	[0, 2.73]	[0, 1.71]	[0, 5.48]	[0, 4.29]
		$\pi^\pm$	1.07	0.83	1.09	0.78	[0, 2.40]	[0, 1.70]	[0, 5.63]	[0, 4.22]
		$\mu^\pm$	0.94	0.71	1.04	0.66	[0, 1.80]	[0, 1.33]	[0, 4.57]	[0, 3.81]
		$e^\pm$	1.18	1.07	1.23	1.06	[0, 2.74]	[0, 2.44]	[0, 6.81]	[0, 5.36]
RNN	**all hits**	all	1.13		1.25		[0, 2.31]		[0, 7.54]	
			stopping	escaping	stopping	escaping	stopping	escaping	stopping	escaping
		“	1.27	1.03	1.50	1.02	[0, 2.79]	[0, 1.96]	[0, 9.03]	[0, 5.95]
		p	1.26	0.99	1.48	0.98	[0, 2.92]	[0, 1.86]	[0, 9,52]	[0, 5.48]
		$\pi^\pm$	1.20	1.04	1.35	0.99	[0, 2.47]	[0, 1.94]	[0, 8.14]	[0, 5.57]
		$\mu^\pm$	1.12	0.95	2.48	0.90	[0, 2.20]	[0, 1.72]	[0, 12.49]	[0, 5.15]
		$e^\pm$	1.45	1.35	1.51	1.36	[0, 3.16]	[0, 2.84]	[0, 9.06]	[0, 8.02]
SIR-PF	**track hits**	all	1.40		1.50		[0, 3.35]		[0, 7.98]	
			stopping	escaping	stopping	escaping	stopping	escaping	stopping	escaping
		“	1.53	1.30	1.69	1.35	[0, 3.70]	[0, 3.08]	[0, 9.28]	[0, 6.91]
		p	1.38	1.19	1.39	1.16	[0, 3.45]	[0, 2.98]	[0, 6.36]	[0, 5.49]
		$\pi^\pm$	1.46	1.29	1.74	1.26	[0, 3.59]	[0, 3.04]	[0, 9.45]	[0, 6.05]
		$\mu^\pm$	1.24	1.19	1.22	1.22	[0, 2.76]	[0, 2.73]	[0, 5.87]	[0, 6.25]
		$e^\pm$	1.92	1.79	1.99	1.78	[0, 4.29]	[0, 3.94]	[0, 11.85]	[0, 9.87]
	all hits	all	2.21		2.00		[0, 3.88]		[0, 10.74]	
			stopping	escaping	stopping	escaping	stopping	escaping	stopping	escaping
		“	2.33	2.13	2.34	1.72	[0, 4.19]	[0, 3.68]	[0, 12.30]	[0, 9.54]
		p	2.33	2.14	2.21	1.83	[0, 4.33]	[0, 3.84]	[0, 12.37]	[0, 10.08]
		$\pi^\pm$	2.23	2.15	2.35	1.72	[0, 3.90]	[0, 3.73]	[0, 11.80]	[0, 9.30]
		$\mu^\pm$	2.18	2.03	3.53	1.56	[0, 3.82]	[0, 3.41]	[0, 21.26]	[0, 8.57]
		$e^\pm$	2.51	2.50	2.24	2.16	[0, 4.59]	[0, 4.63]	[0, 12.43]	[0, 11.37]

TABLE I. Euclidean distance (mean μ, standard deviation σ, and ranges for the one-sided 68% and 95% areas) between the predicted and the true nodes for the Transformer, recurrent neural network (RNN), and the Sequential Importance Resampling particle filter (SIR-PF) algorithms. For the latter, the table shows the results after inputting: (1) all the hits, (2) track hits only. It also shows the results independently for each particle type (proton p, pion $\pi^\pm$, muon $\mu^\pm$, and electron $e^\pm$) and distinguishes whether the particle escaped or stopped at the detector.

and accomplish spatial determination <1.5 mm far (on average) from the real physical case.

A more exhaustive analysis of the performance of both methods is presented in Tab. I, which reveals the effec-

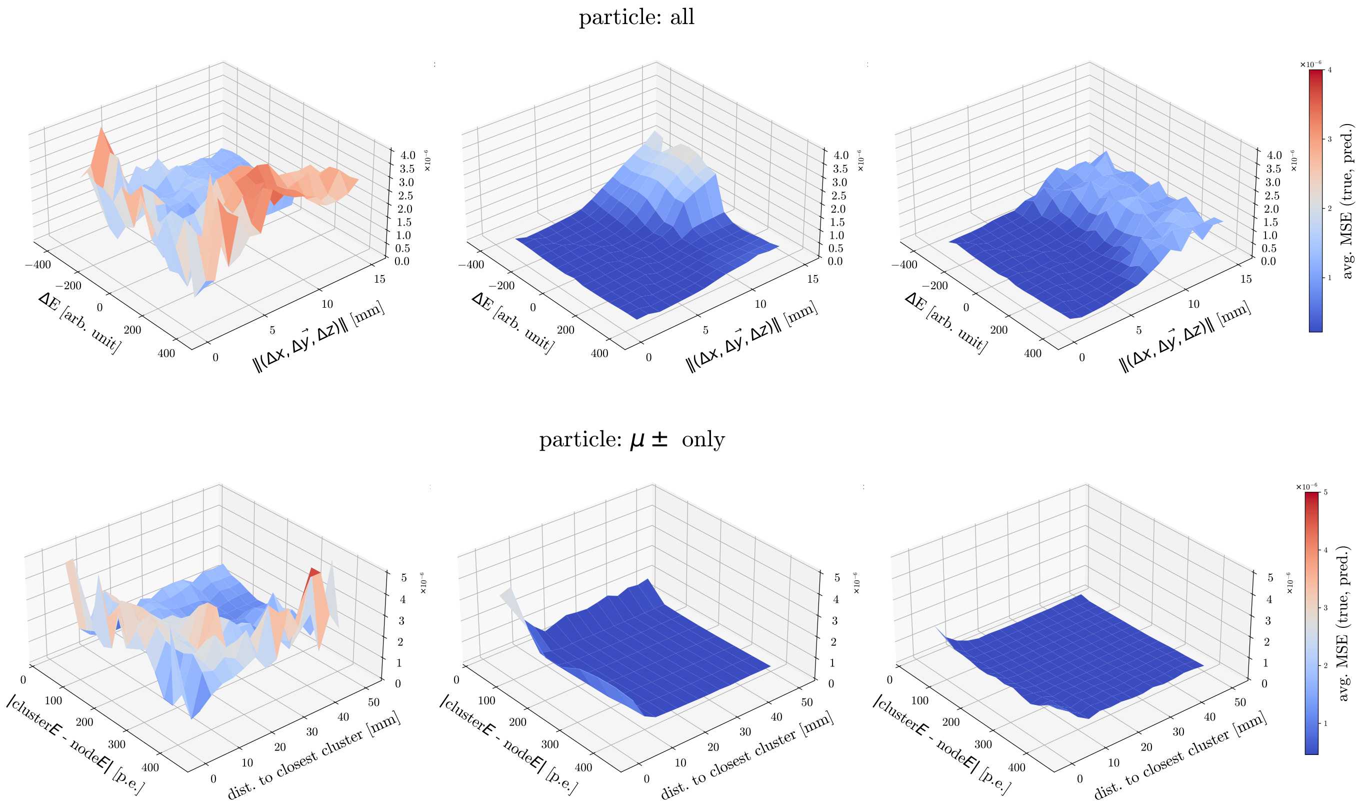

FIG. 3. (Top) Behaviour of the mean-squared-error (MSE) loss concerning $\|(\Delta x, \vec{\Delta y}, \Delta z)\|$ (magnitude of the vector resulting from the differences in position between consecutive nodes) and ΔE (differences in energy deposition between successive nodes) for the three algorithms: Sequential Importance Resampling particle filter (SIR-PF) with all hits (left), recurrent neural network (RNN, middle), and Transformer (right). After standardisation, each bin corresponds to the average mean-squared error (MSE) loss applied to the pair (true node, fitted/predicted node). All fitted nodes are considered. (Bottom) Behaviour of the mean-squared-error (MSE) loss concerning the distance from each fitted node to the closest cluster hit and $|clusterE - nodeE|$ (absolute difference between the energy depositions of the fitted node and the nearest cluster hit) for the three algorithms: SIR-PF with all hits (left), RNN (middle), and Transformer (right). After standardisation, each bin corresponds to the average mean-squared error (MSE) loss applied to the pair (true node, fitted/predicted node). Only nodes from muon ($\mu^{\pm}$) particles are considered.

tiveness of the NNs compared to the SIR-PF variants. The table also confirms that the track fitting becomes more manageable when the crosstalk hits are removed from the input and more precise information is given to the filter (the SIR-PF version with only track hits outperforms the one with all hits as input). This last fact also evidences the power of deep learning, which is, on average, able to predict more accurately the node positions and thus the true track trajectory, even if its input consists of all the reconstructed hits without any type of pre-processing (e.g., removal of crosstalk hits), meaning that it could understand the relations between hits internally, confirming the ability to discard the crosstalk hits during the fitting calculation. In order to compare the Transformer and the RNN, it is worth looking at the muon fitting at Tab. I: the Transformer reports the best results for fitting muon particles (for both mean and standard deviation) in contrast to the RNN, which reports an atypically large std dev. for muon tracks contained in the detector. The explanation relies on the length of the particles and the properties of the algorithms: since muons tend to have the most extended track length among the simulated stopping particles (protons and pions tend to have more secondary interactions and electrons produce electromagnetic showers), and the RNN depends on its memory mechanisms to bring features from faraway hits to fit a particular reconstructed hit (see Sec.VII, Supplementary Information, for more details), it is habitual to omit some information from remote hits during the fitting; on the other hand, the Transformer reduces its mistakes by having a complete picture of the particle thanks to its capacity to learn the correlations among all reconstructed hits.

To understand the behaviour of the fittings for the different physical structures of the particles, we have calculated the mean-squared error (MSE, which is the loss function used during the neural network trainings) between each fitted and true node and visualised the information in Fig. 3. The MSE loss, which penalises outliers by construction, seems flatter for the RNN and Transformer than for the SIR-PF, indicating more stability in the fitting. Besides, it is notorious for highlighting the tendency for particular negative ΔE values to report high losses in the NN cases, caused mainly due to the low charge of crosstalk compared to track hits. Besides, Fig. 3, as expected, also reveals that the three algorithms report worse fittings when getting closer to cluster hits connected to the track. For instance, in the case of muon particles, these clusters are typically due to the ejection of δ-rays, i.e. orbiting electrons knocked out of atoms, often causing a kink on the muon track; however, both NNs seem to deal much better with this attribute.

Even if the primary goal of this article is to show the performance of the fitting from a physics perspective, it is worth comparing the different algorithms in terms of computing time. Table II manifests the average time it takes for each algorithm to run the fitting on a single particle. The results exhibit a considerable speedup for both the RNN and the Transformer models (with speedups of $\sim \times 4$ and $\sim \times 35$, respectively) with a single thread on the CPU. The table does not show the SIR-PF results for the distributed computing scenarios since it would require some time to parallelise the SIR-PF code to run it with multiple threads or to adapt it to GPU computation, which is clearly beyond the scope of the study; that being said, the table shows the parallel results for the RNN and Transformer cases since these are features available in the PyTorch framework, which show how inexpensive it would be to achieve significant speedups for an ordinary user.

Processor	Parallelisation	SIR-PF	RNN	Transformer
CPU	single-thread	435.71 ± 5.18	91.16 ± 1.17	12.25 ± 0.19
	multi-thread	-	82.22 ± 1.00	6.58 ± 0.04
GPU	batch_size = 1	-	31.27 ± 0.99	8.96 ± 0.31
	batch_size = 16	-	4.02 ± 0.12	1.24 ± 0.12
	batch_size = 64	-	1.43 ± 0.05	0.71 ± 0.04

TABLE II. Average computing time each algorithm takes to process a single particle (in milliseconds). The test shows the average results of running the three methods (Sequential Importance Resampling particle filter (SIR-PF) with all hits, recurrent neural network (RNN), and Transformer) on the same ten random subsets of the testing dataset consisting of 10,000 particles each. CPU: AMD EPYC 7742 64-Core 3200MHz Processor, GPU: NVIDIA A100 Tensor Core (8GB of memory). Note that the SIR-PF implemented does not support multi-threading nor GPU computation since it is out of the scope of the article; parallelising the computation for the RNN and Transformer becomes trivial thanks to PyTorch. The parameter "batch_size" indicates the number of particles processed together in each step.

Finally, if we look at the size of the histogram used to calculate the likelihood, it consists of 3,948,724 bins with non-zero values, compared to the 213,553 learnt parameters of the RNN ($\sim$18 times fewer parameters) and the 167,875 parameters of the Transformer ($\sim$23 times fewer parameters than the SIR-PF histogram). Of course, it would be possible to design a more efficient version of the histogram (which is also out of the scope of the article) to reduce the difference in parameters among the methods. Nevertheless, this first approximation already gives insights into how compact the information is encoded in the neural network cases in contrast to the Bayesian filter scenario with a physics-based likelihood calculation.

B. Impact on the detector physics performance

The reconstruction of the primary particle kinematics provides diverse information: the electric charge (negative or positive); the identification of the particle type (protons, pions, muons, electrons), which mainly depends on the particle stopping power as a function of its momentum; the momentum, either from the track range of the particle that stops and releases all its energy in the detector active volume or from the curvature of its track if the detector is immersed in a magnetic volume; the direction. An improved resolution on the spatial coordinate and, consequently, of the particle stopping power impacts the accuracy and precision of the physics measurement. This section compares the performance of the reconstruction of particle interactions provided by the Transformer and RNN to the one using the SIR-PF.

The charge identification (charge ID) is performed by reconstructing the curvature of the particle track in the detector immersed in the 0.5 T magnetic field. The charge ID performance was studied for muons (resp. electrons) with momenta between 0 and 2.5 GeV/c (resp. 0 and 3.5 GeV/c) and isotropic direction distribution. From Fig. 4, it is evident that the NNs outperform the SIR-PF. For instance, the muon charge can be identified with an accuracy better than 90% if the track has a length projected on the plane transverse to the magnetic field of $\sim$33 and $\sim$36 cm for the Transformer and RNN, respectively. Instead, the SIR-PF (with all the hits, the version with the same input as the neural network cases) requires a track of at least $\sim$42 cm in order to achieve the same performance. Similar conclusions can be derived from the charge ID study on electrons and positrons.

In Fig. 4, the case of a 0.6 GeV/c muon was also studied, showing the node positions fitted with the NNs and SIR-PF, with the Transformer better capturing the curvature due to the magnetic field. It was found that if the tracking resolution is accurate, it is possible to either improve the detector performance beyond its design or to aim for a more compact design of the scintillator detector deployed in a magnetic field. For instance, the spatial resolution achieved with the NNs in a magnetic field of 0.5 T allows measuring the momentum of a 0.6 GeV/c muon from its curvature with a resolution of about 15% with a length of the track projected on the plane transverse to the magnetic field of almost 40 cm, shorter by about 20 cm than the length needed by the SIR-PF with

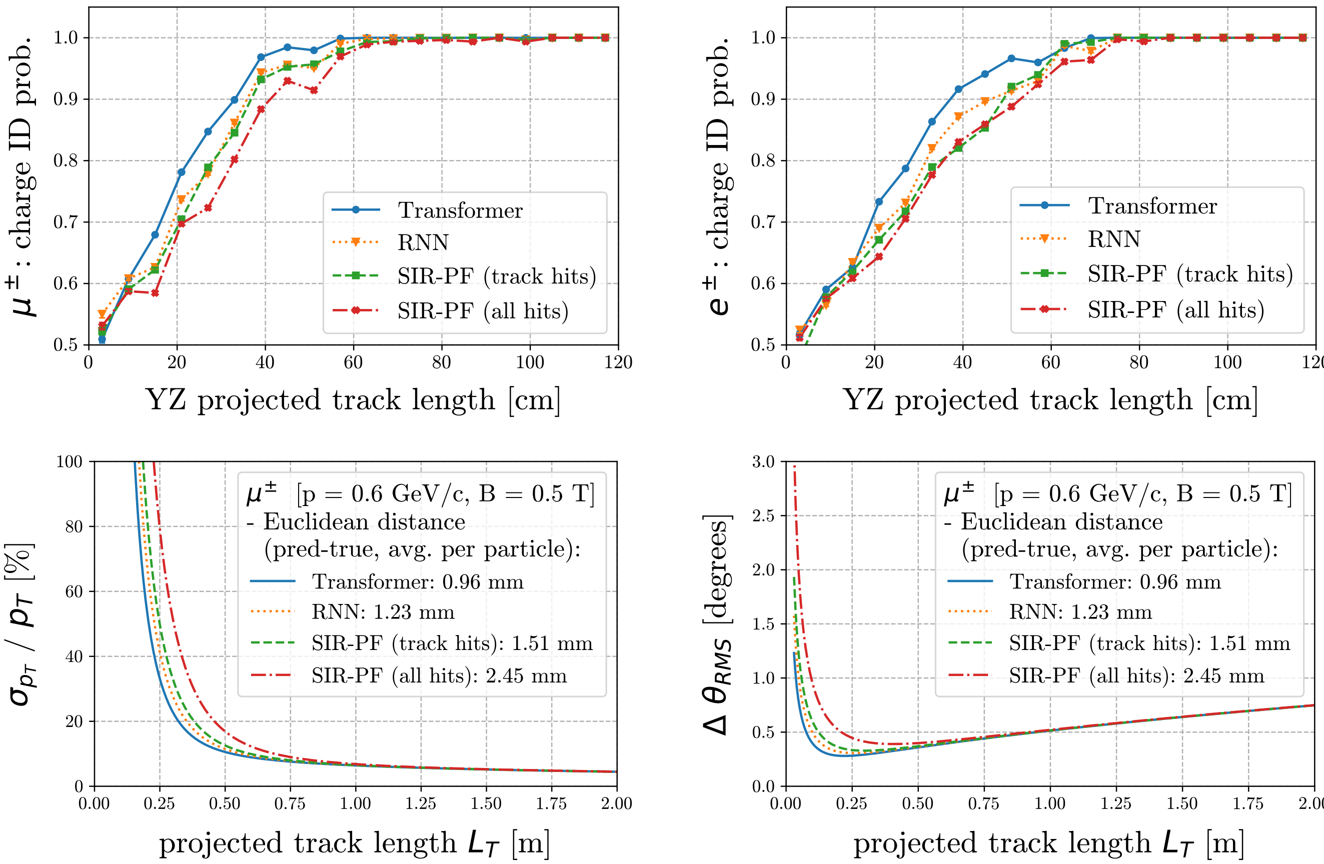

FIG. 4. (Top) Charge ID probability for muons and antimuons ($\mu^\pm$, left) and electrons and positrons ($e^\pm$, right) as a function of the track length projected on the plane perpendicular to the 0.5 T magnetic field. An equal number of particles and antiparticles are considered in both cases. (Bottom) A muon example of 0.6 GeV/c with a 0.5 T magnetic field is considered to show the momentum-by-curvature resolution as a function of the track length projected on the plane perpendicular to the magnetic field (left), as well as the angular resolution as a function of the particle length in the detector. The average Euclidean distance (between true and fitted nodes) per muon particle was considered, and the results are presented for the different fitting techniques: Transformer, recurrent neural network (RNN), and Sequential Importance Resampling particle filter (SIR-PF) with all hits and only track hits as input.

all the hitsO. Such an improvement implies the possibility of accurately reconstructing the momentum of muons escaping the detector for a larger sample of data. At the same time, improved methods for the reconstruction of particle interactions could become a new tool in the design of future particle physics experiments, for example leading to more compact detectors, thus lower costs. Similar conclusions can be achieved about the particle angular resolution, improved by about a factor of two and, simultaneously, requiring a track length three times shorter than the one obtained with traditional methods.

The Transformer outperforms the SIR-PF also in the reconstruction of the particle momentum, both by range and curvature. For instance, the momentum-by-range resolution for protons stopping in the detector between 0.9 and 1.3 GeV/c is improved by a factor of ∼15%, as shown in Fig. 5. Since protons typically have a much stronger stopping power towards the end of the track (Bragg peak), the total amount of energy leaked to the adjacent cubes is more significant. We observe that the fitting near the Bragg peak becomes more challenging for protons (for example, compared to muons) and less precise due to the presence of more crosstalk hits. This becomes particularly relevant for low momentum (true initial momentum from 0.4 to 0.8 GeV/c) - hence short - protons. However, the Transformer seems to deal well with this difficulty, whilst the RNN reports worse resolutions for this particular case, as shown in Fig. 5.

The particle identification performance depends on the capability of reconstructing the particle stopping power along its path as a function of its initial momentum. The resolution to the particle dE/dx is shown in Fig. 5, where one can see that the energy deposited by a proton as a function of the fitted node position is neater and more refined for the NNs compared to the SIR-PF (with all hits as input), in particular for the Transformer that shows the most accurate Bragg peak. Automatically, this translates into a more performing

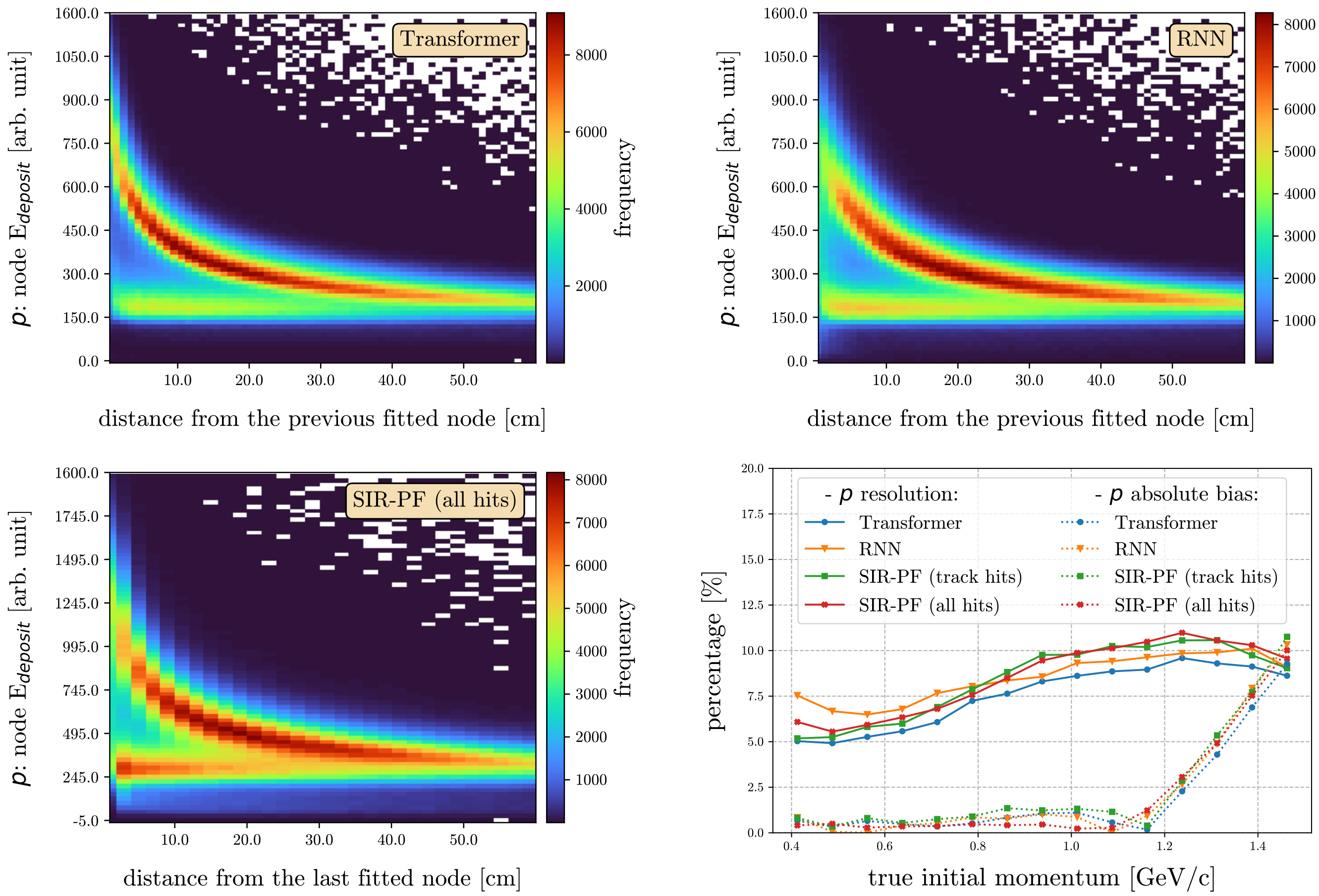

FIG. 5. Measured energy deposited by a stopping proton at each fitted node as a function of its distance from the last fitted node for the Transformer (top left), the recurrent neural network (RNN, top right), and Sequential Importance Resampling particle filter (SIR-PF) with all hits as input (bottom left). Note that we chose a different binning for the SIR-PF than the one used for the NN versions for visualisation reasons since the former algorithm reports fewer fitted nodes per particle on average. (Bottom right) The reconstructed momentum bias (dashed line) and resolution (solid line) for stopping protons as a function of real initial proton momentum are shown for the different fitting algorithms.

particle identification capability, as shown in Tab. III for different particles such as muons, pions, protons and electrons for a wide range of energies.

IV. DISCUSSION

Deep learning is starting to play a more relevant role in the design and exploitation of particle physics experiments, although it is still in a gestation phase within the high-energy physics community. If the optimal neural network is optimised, deep learning has the unique capability of building a non-linear multi-dimensional MC-based prior probability function with many degrees of freedom (d.o.f.) that can efficiently and accurately model all the information acquired in a particle physics experiment and enhance the performance of the particle track fitting and, consequently, its kinematics reconstruction. Such a level of detail is, otherwise, nearly impossible to incorporate "by hand" in the form of, for example, a covariance matrix to be used in a traditional particle filter. In this work, we show that a Transformer and a RNN can efficiently learn the details of the particle propagation in matter mixed with the detector response and lead to a significantly improved reconstruction of the interacting particle kinematics. We observed that the NNs capture better the details of the particle propagation even when its complexity increases, which is the case near the presence of clusters of hits, for example, due to δ-rays.

It is worth noting that, as mentioned in Sec. III, this work does not aim to report on the performance of the simulated particle detector but rather to show the added value provided by a NN-based fitting. Moreover, the proposed method does not replace the entire chain of algorithms traditionally adopted in a particle flow analysis (e.g., minimum spanning tree, vertex fitting, etc.) but is meant to assist and complement them as a more performing fitter. For instance, a possibility could be to apply SIR-PF several times with "ad-hoc" manipulation of the data between each step. However, this would be an unfair comparison as one could also implement multiple

		Truth			
		p	$\pi^{\pm}$	$\mu^{\pm}$	$e^{\pm}$
Transformer	p	0.907	0.057	0.071	0.020
	$\pi^{\pm}$	0.067	0.643	0.190	0.199
	$\mu^{\pm}$	0.007	0.041	0.595	0.009
	$e^{\pm}$	0.019	0.259	0.144	0.772
RNN	p	0.896	0.080	0.089	0.027
	$\pi^{\pm}$	0.073	0.623	0.233	0.200
	$\mu^{\pm}$	0.006	0.036	0.506	0.007
	$e^{\pm}$	0.025	0.261	0.172	0.766
SIR-PF (track hits)	p	0.858	0.080	0.082	0.017
	$\pi^{\pm}$	0.103	0.606	0.310	0.237
	$\mu^{\pm}$	0.014	0.042	0.453	0.006
	$e^{\pm}$	0.025	0.272	0.155	0.740
SIR-PF (all hits)	p	0.891	0.092	0.126	0.024
	$\pi^{\pm}$	0.077	0.603	0.236	0.229
	$\mu^{\pm}$	0.008	0.039	0.517	0.007
	$e^{\pm}$	0.024	0.266	0.121	0.740

TABLE III. Particle identification (proton p, pion $\pi^{\pm}$, muon $\mu^{\pm}$, and electron $e^{\pm}$) confusion matrix for different methods: RNN, Transformer, Sequential Importance Resampling particle filter (SIR-PF) with all hits, and SIR-PF with only track hits as input. Each matrix element corresponds to the probability of correctly identifying an elementary particle. Each column of the confusion matrix is normalized to 1 and represents the true particles, whereas the rows represent the predictions.

deep learning methods and focus on their optimisation.

We believe this approach is a milestone in artificial intelligence applications in HEP and can play the role of a game changer by shifting the paradigm in reconstructing particle interactions in the detectors. The prior, which is consciously built from the modelling of the underlying physics from data external to the experiment, becomes as essential as the real data collected for the physics measurement. De facto, the prior provides a strong constraint to the "interpretation" of the data, helping to remove outliers introduced by detector effects such as from the smearing introduced by the point spread functionand improving the spatial resolution well below the actual granularity of the detector.

Its accuracy also depends on the quality of the training sample, i.e. on the capability of the MC simulation to correctly reproduce the data. Although this is true for most of the charged particles, a careful characterisation of the detector response will be crucial to validate and, if necessary, tune the simulation (e.g., electromagnetic shower development or hadronic secondary interactions) used to generate the training sample.

This study requires that, first, the signatures observed in the detector are analysed, and the three-dimensional hits that compose tracks belonging to primary particles (directly produced at the primary interaction vertex) are distinguished and analysed independently. This approach is typical of particle flow analyses.

This work is focused on physics exploitation in particle physics experiments. However, the developed AI-based methods can also fulfil the requirements in applications outside of HEP, as long as one has a valid training dataset. One example is proton computed tomography [51–54] used in cancer therapy, where scintillator detectors are used to measure the proton stopping power along its track in the Bragg peak region to precisely predict the stopping position of the proton in the human body. This measurement is analogous to the momentum regression described in Sec. V B, given the nearly complete correlation between the particle range and momentum.

Future improvements to the developed NNs may involve the direct computation of the node stopping power from the track, i.e., the combined fitting of both the node particle position and energy loss.

V. METHODS

A. Description of the fitting algorithms

To test the capability of deep learning to fit particle trajectories using reconstructed hits as input, we developed two neural networks that represent the state-of-the-art in the field of natural language processing (NLP, as detailed in the Supplementary Information): the recurrent neural network (RNN) [47–49] and the Transformer [50] (see Fig. 6 for a full picture of the architectures). Both algorithms learn from input sequences, each of these sequences being, for instance, a succession of words forming a sentence in the NLP case; or reconstructed hits representing a detected elementary particle in our scenario. Their power rely on their capacity of learning relations between all elements of a sequence. In general terms, RNNs count with memory mechanisms to use information from the "past" (previous items in the sequence) and the "future" (following items in the sequence) to make predictions. Thus, RNNs assume the input sequences to be ordered. On the other hand, Transformers do not necessarily need sequences to be ordered: the correlations among different items in the sequence are learnt throughout the training process.

We implemented a bi-directional RNN, and the memory mechanism used is the gated recurrent unit (GRU) [55]. Our RNN consists of five bi-directional GRU layers with 50 hidden units each. The output of each GRU layer is the concatenation of the forward and backward modules of the layer and is given as input for the following layer (except for the last layer). Instead of propagating only the output of the last GRU layer to the final dense layer, the outputs of all layers are summed together, replicating the concept of "skipped connections" in a similar way to what the ResNet or DenseNet model do [56]. As regularisation, a dropout of 0.1 is applied to the output of each GRU layer (except for the last GRU layer) and to the summed output of the GRU layers,

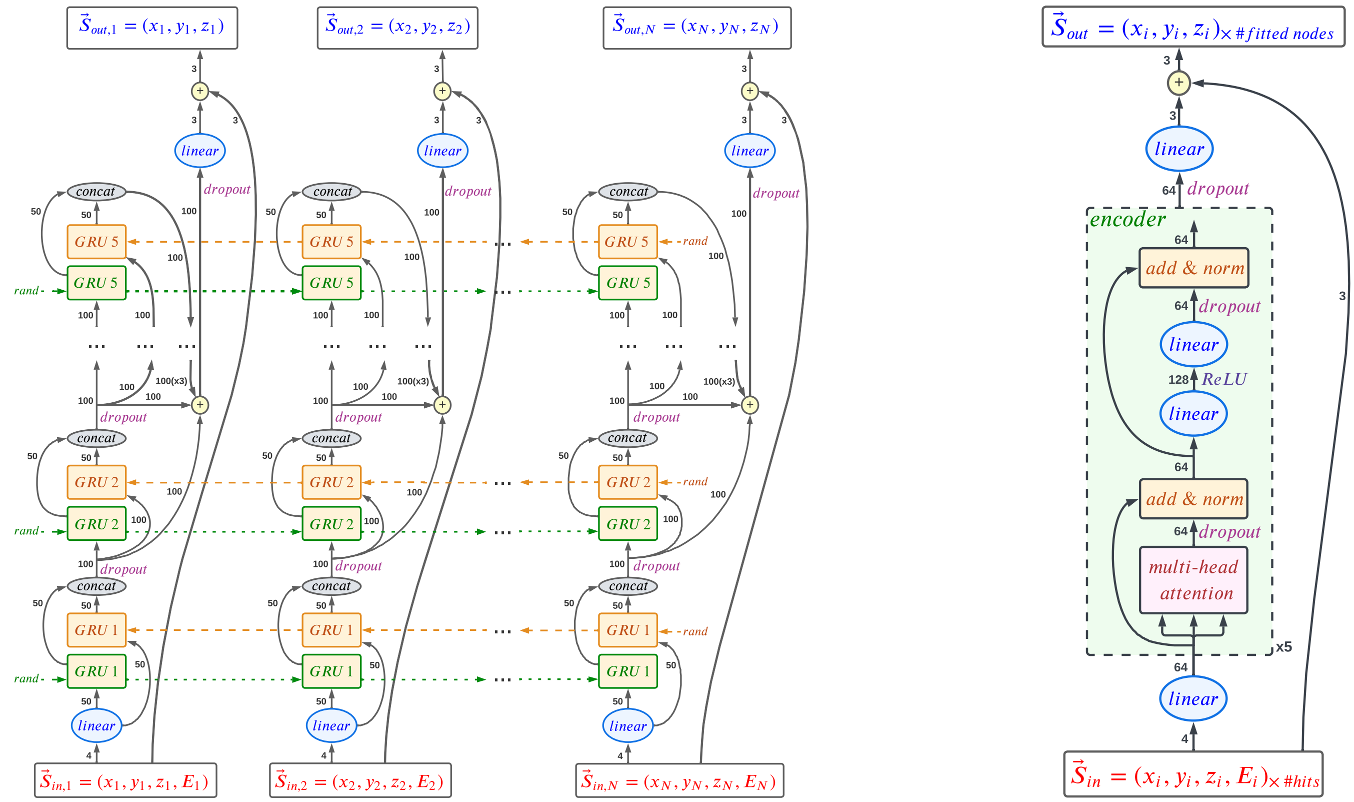

FIG. 6. The architectures of the neural networks implemented: recurrent neural network (RNN, left) and Transformer (right). In high-level terms, RNN consists of five bidirectional GRU layers, while the Transformer consists of five sub-encoder layers. Both models are followed by a linear layer that projects the sum of the outputs of the GRU/encoder layers into a vector of length three. Finally, the input hit position (x_i, y_i, z_i) is summed to the network's output, allowing it only to learn the "residuals" of the reconstructed hits concerning the true node states ($\vec{S}_{in} \to \vec{S}_{out}$).

which is then projected through a final dense layer to have fitted nodes of size 3, representing the coordinates in a three-dimensional space (x, y, and z). The implemented RNN has a total of 213,553 trainable parameters.

The Transformer model designed consists of 5-stacked Transformer-encoder layers, with 8 heads per layer and a dimension of 128 for the hidden dense layer. The input hits are embedded into vectors of size 64. A dropout of 0.1 is applied in each encoder layer and also to the output of the encoder layers to be further projected through a final dense layer (analogously to the RNN), making each fitted node have a length of three. There is no positional encoding since the goal is to make the network learn the relative ordering of the hits based on the 3D positions. The network has a total of 167,875 trainable parameters.

We implemented both networks in Python v3.10.4 [57] using PyTorch version 1.11.0 [58], and trained them on a dataset of simulated elementary particles consisting of 1,762,327 particles (414,824 protons, 432,855 pions, 446,858 muons and antimuons, and 467,790 electrons and positrons). Each particle consists of a sequence of reconstructed hits with their known positions (centre of the matching cubes) and energy depositions (in an arbitrary signal unit) represented for each hit with the tuple $\vec{S}_{in} = (x_i,\ y_i,\ z_i,\ E_i)$ and truth node position to be learnt $\vec{S}_{out} = (x_i,\ y_i,\ z_i)$. Each variable is normalised to the range [0,1]. We used 80% of the particles from this sample for training and 20% for validation, ignoring particles with either less than 10 reconstructed hits or less than 2 track hits, both representing less than 1% of the total particles. Note that this dataset is statistically independent of the one used for producing the results shown in Sec. III. Mean-squared error and Adam (batch size of 128, learning rate of 10^{-4}, $\beta_1 = 0.9$, and $\beta_2 = 0.98$) are the loss function (typical for regression) and optimiser, respectively, chosen for both networks. We trained the models on an NVIDIA A100 GPU for an indefinite number of epochs but with an early stopping of 30, meaning that the training terminates when the loss on the validation set does not improve for 30 epochs. The training and validation losses are shown in Fig. 7.

It is necessary to mention that for both the RNN and the Transformer, we sum together (position-wise) the output of the models for each fitted node and the 3D position of the corresponding reconstructed hit given as input. In that way, we force the networks to learn the residuals between reconstructed hits and fitted nodes (in other words, what is learnt is how to adjust each reconstructed hit to a node position that matches the actual particle trajectory).

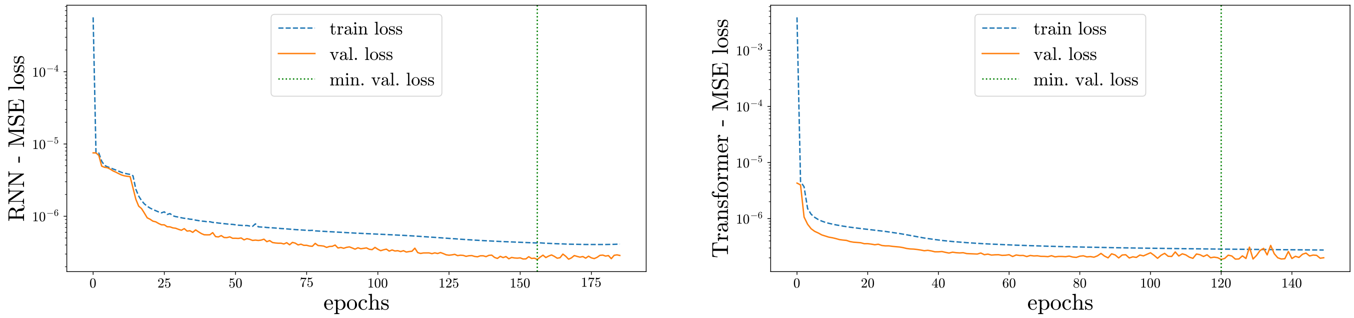

FIG. 7. Training and validation loss curves for the recurrent neural network (RNN, left) and the Transformer (right). The loss function used is the mean-squared error (MSE). The dashed-vertical lines represent the epoch that minimises the loss and, thus, the model weights used for the subsequent analysis. The Transformer network converges much faster than the RNN, presumably because the former can learn the correlations among unordered reconstructed hits, and the latter assumes the reconstructed hits are ordered, which can lead to confusion due to the inherent flaws of the ordering provided (impossibility of arranging an optimal order from reconstructed information)

Regarding the Sequential Importance Resampling particle filter (SIR-PF), for each particle, we use the first reconstructed hit as prior[ii], meaning we use it to sample the first random particles inside that cube, and the energy deposition of each particle happens to be the one of the hitting cube. In each step, the random particles are propagated through the next 15 hits[iii] (starting with counting from the position of the current state). For each random particle, the algorithm calculates the variation in x, y, z, θ (elevation angle defined from the XY-plane, in spherical coordinates), and energy deposition (in an arbitrary signal unit) between the particle and the current state and assigns a likelihood based on the value of the selected bin in a 5-dimensional histogram[iv], pre-filled using the same dataset used to train the RNN and the Transformer. In that way, the next state ends up being the weighted average (using the pre-computed likelihood) of the positions of the different sampled particles available. The filter is run from the start to the end of the particle (forward fitting) and from the end to the start (backward fitting); the results of the forward and backward fittings are averaged in a weighted manner, giving more relevance to nodes fitted last in both cases. The total number of random particles sampled in each step is 10,000.

[ii] Hits are reordered with respect to the axis the particle is travelling through the furthest; if there are several candidates for the first position, we chose the one with the highest energy deposition.

[iii] We make sure the random particles are sampled inside the available reconstructed hits.

[iv] The histogram, used for the likelihood calculation of the SIR-PF, is filled with the variation between consecutive true nodes in x, y, z, θ, and energy deposition, named: Δx, Δy, Δz, $\Delta\theta$, and ΔE, respectively. The histogram has 100 bins per dimension.

B. Computation of particle kinematics

The RNN, Transformer, and SIR-PF outputs are analysed to extract the kinematics from the fitted tracks. The performance of the methods depends on the accuracy of the fitted nodes compared to the true track trajectories. The same procedure has been applied to the nodes fitted with the different algorithms for a fair comparison.

The following steps have been followed to perform the physics analysis, that is, particle identification (PID), momentum reconstruction and charge identification (charge ID):

1. Extract "track" nodes: the input 3D hits can be divided into two categories: (1) track hits, directly crossed by the charged particle, (2) crosstalk hits, caused by the leakage of scintillation light from the cube containing the charged particle. After the track is fitted, the 3D hits are identified as track-like if there is a scintillator cube with a particular energy deposition that contains the fitted node. The remaining nodes are classified as non-track, and they include crosstalk hits. The scintillation light observed in a non-track hit is summed to the nearest track hit. The position of the fitted node is then used to compute the stopping power ($\mathrm{d}E/\mathrm{d}x$).

2. Node energy smoothing: the energy of the remaining "track" nodes is smoothed in order to eliminate fluctuations due, for example, to the different path lengths travelled by the particle in the adjacent cubes (the scintillation light in a cube is nearly proportional to the distance travelled by the particle). The smoothing of an energy node is performed by applying an average over the energy of nearby nodes weighted by a Gaussian distribution function of the respective distance.

3. Particle identification and momentum regression: a gradient-boosted decision tree (GBDT) [59], avail-

able in the TMVA package of the CERN ROOT analysis software (`https://root.cern.ch/`), was used to perform the particle identification and the momentum regression. The GBDT input parameters were chosen as: (1) the first 5 and the last 10 fitted node energies along the track; (2) the neighbouring node distances of those 15 nodes; (3) the track total length and energy deposition. Two independent GBDTs with the same structure were trained to reconstruct the primary particle type (muon, proton, pion, or electron, classification) and its initial momentum (regression).

The electric charge of the particle was identified by measuring the deflection of the track projected to the plane perpendicular to the magnetic field. The convex or concave deflection implies either a positive or a negative charge, where the positions of the fitted nodes were used.

The momentum reconstruction from the track curvature produced by the magnetic field was estimated for the resolutions provided by different track fitters and studied for different configurations by using parameterised formulas that incorporate the spatial resolution from tracking in a magnetic field as well as the multiple scattering in dense material [60, 61], that have been shown to reproduce data well enough for sensitivity studies.

VI. ACKNOWLEDGEMENTS

Part of this work was supported by the SNF grant PCEFP2_203261, Switzerland.

---

[1] F. Hasert, H. Faissner, W. Krenz, J. Von Krogh, D. Lanske, J. Morfin, K. Schultze, H. Weerts, G. Bertrand-Coremans, J. Lemonne, *et al.*, "Search for elastic muon-neutrino electron scattering," *Physics Letters B*, vol. 46, no. 1, pp. 121–124, 1973.

[2] F. Hasert, S. Kabe, W. Krenz, J. Von Krogh, D. Lanske, J. Morfin, K. Schultze, H. Weerts, G. Bertrand-Coremans, J. Sacton, *et al.*, "Observation of neutrino-like interactions without muon or electron in the Gargamelle neutrino experiment," *Nuclear Physics B*, vol. 73, no. 1, pp. 1–22, 1974.

[3] S. Chatrchyan, V. Khachatryan, A. Sirunyan, A. Tumasyan, W. Adam, E. Aguilo, T. Bergauer, M. Dragicevic, J. Erö, C. Fabjan, *et al.*, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," *Physics Letters B*, vol. 716, no. 1, pp. 30–61, 2012.

[4] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. Abdel Khalek, A. Abdelalim, O. Abdinov, R. Aben, B. Abi, M. Abolins, *et al.*, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," *Physics Letters B*, vol. 716, no. 1, pp. 1–29, 2012.

[5] F. Abe, H. Akimoto, A. Akopian, M. G. Albrow, S. R. Amendolia, D. Amidei, J. Antos, C. Anway-Wiese, S. Aota, G. Apollinari, *et al.*, "Observation of top quark production in $\overline{p}p$ collisions with the Collider Detector at Fermilab," *Phys. Rev. Lett.*, vol. 74, pp. 2626–2631, Apr 1995.

[6] L. Gruber, "LHCb SciFi — upgrading LHCb with a scintillating fibre tracker," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 958, p. 162025, 2020. Proceedings of the Vienna Conference on Instrumentation 2019.

[7] S. Amerio, S. Amoruso, M. Antonello, P. Aprili, M. Armenante, F. Arneodo, A. Badertscher, B. Baiboussinov, M. Baldo Ceolin, G. Battistoni, *et al.*, "Design, construction and tests of the ICARUS T600 detector," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 527, no. 3, pp. 329–410, 2004.

[8] B. Abi *et al.*, "Deep Underground Neutrino Experiment (DUNE), Far Detector Technical Design Report, Volume II DUNE Physics," 2 2020.

[9] R. Acciarri *et al.*, "Design and Construction of the MicroBooNE Detector," *JINST*, vol. 12, no. 02, p. P02017, 2017.

[10] A. Blondel, F. Cadoux, S. Fedotov, M. Khabibullin, A. Khotjantsev, A. Korzenev, A. Kostin, Y. Kudenko, A. Longhin, A. Mefodiev, *et al.*, "A fully-active fine-grained detector with three readout views," *Journal of Instrumentation*, vol. 13, pp. P02006–P02006, feb 2018.

[11] V. Andreev *et al.*, "A high-granularity plastic scintillator tile hadronic calorimeter with APD readout for a linear collider detector," *Nucl. Instrum. Meth. A*, vol. 564, pp. 144–154, 2006.

[12] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 03 1960.

[13] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, pp. 107–113(6), April 1993.

[14] V. Innocente, M. Maire, and E. Nagy, "GEANE: Average tracking and error propagation package," in *Workshop on Detector and Event Simulation in High-energy Physics (MC '91)*, pp. 58–78, 1991.

[15] V. Innocente and E. Nagy, "Trajectory fit in presence of dense materials," *Nucl. Instrum. Meth. A*, vol. 324, pp. 297–306, 1993.

[16] A. Cervera-Villanueva, J. J. Gomez-Cadenas, and J. A. Hernando, "'RecPack' a reconstruction toolkit," *Nucl. Instrum. Meth. A*, vol. 534, pp. 180–183, 2004.

[17] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, *et al.*, "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[18] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce Dubois, M. Asai, G. Barrand, R. Capra, S. Chauvie, R. Chytracek, *et al.*, "Geant4 developments and applications," vol. 53, no. 1, pp. 270–278, 2006.

[19] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso, E. Bagli, A. Bagulya, S. Banerjee, G. Barrand, *et al.*, "Recent developments in Geant4," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 835, pp. 186–225, 2016.

[20] C. Ahdida *et al.*, "New Capabilities of the FLUKA Multi-Purpose Code," *Front. in Phys.*, vol. 9, p. 788253, 2022.

[21] G. Battistoni *et al.*, "Overview of the FLUKA code," *Annals Nucl. Energy*, vol. 82, pp. 10–18, 2015.

[22] L. de Oliveira, M. Paganini, and B. Nachman, "Learning particle physics by example: location-aware generative adversarial networks for physics synthesis," *Computing and Software for Big Science*, vol. 1, no. 1, pp. 1–24, 2017.

[23] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, "Machine learning at the energy and intensity frontiers of particle physics," *Nature*, vol. 560, no. 7716, pp. 41–48, 2018.

[24] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to LHC physics," *Annual Review of Nuclear and Particle Science*, vol. 68, no. 1, pp. 161–181, 2018.

[25] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, p. 045002, Dec 2019.

[26] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, vol. 5, no. 1, pp. 1–9, 2014.

[27] B. Abi, R. Acciarri, M. Acero, G. Adamov, D. Adams, M. Adinolfi, Z. Ahmad, J. Ahmed, T. Alion, S. Alonso Monsalve, and et al., "Neutrino interaction classification with a convolutional neural network in the DUNE far detector," *Physical Review D*, vol. 102, Nov 2020.

[28] F. Drielsma, K. Terao, L. Dominé, and D. H. Koh, "Scalable, end-to-end, deep-learning-based data reconstruction chain for particle imaging detectors," 2021.

[29] M. Andrews, M. Paulini, S. Gleyzer, and B. Poczos, "End-to-end physics event classification with cms open data: Applying image-based deep learning to detector data for the direct classification of collision events at the LHC," *Computing and Software for Big Science*, vol. 4, no. 1, pp. 1–14, 2020.

[30] S. Alonso-Monsalve, D. Douqa, C. Jesús-Valls, T. Lux, S. Pina-Otey, F. Sánchez, D. Sgalaberna, and L. H. Whitehead, "Graph neural network for 3D classification of ambiguities and optical crosstalk in scintillator-based neutrino detectors," *Phys. Rev. D*, vol. 103, p. 032005, Feb 2021.

[31] A. Aurisano *et al.*, "A convolutional neural network neutrino event classifier," *Journal of Instrumentation*, vol. 11, no. 09, p. P09001, 2016.

[32] T. Q. Nguyen, D. Weitekamp, D. Anderson, R. Castello, O. Cerri, M. Pierini, M. Spiropulu, and J.-R. Vlimant, "Topology classification with deep learning to improve real-time event selection at the LHC," *Computing and Software for Big Science*, vol. 3, no. 1, pp. 1–14, 2019.

[33] S. Bhattacharya, S. Nandi, S. K. Patra, and S. Sahoo, "'deep' dive into $b \to c$ anomalies: Standardized and future-proof model selection using self-normalizing neural networks," 2020.

[34] P. Abratenko, M. Alrashed, R. An, J. Anthony, J. Asaadi, A. Ashkenazi, S. Balasubramanian, B. Baller, C. Barnes, G. Barr, *et al.*, "Semantic segmentation with a sparse convolutional neural network for event reconstruction in MicroBooNE," *Phys. Rev. D*, vol. 103, p. 052012, Mar 2021.

[35] S. Cheong, A. Cukierman, B. Nachman, M. Safdari, and A. Schwartzman, "Parametrizing the detector response with neural networks," *Journal of Instrumentation*, vol. 15, pp. P01030–P01030, jan 2020.

[36] Z. Qian, V. Belavin, V. Bokov, R. Brugnera, A. Compagnucci, A. Gavrikov, A. Garfagnini, M. Gonchar, L. Khatbullina, Z. Li, *et al.*, "Vertex and energy reconstruction in juno with machine learning methods," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1010, p. 165527, 2021.

[37] K. Carloni, N. W. Kamp, A. Schneider, and J. M. Conrad, "Convolutional neural networks for shower energy prediction in liquid argon time projection chambers," *Journal of Instrumentation*, vol. 17, no. 02, p. P02022, 2022.

[38] R. Acciarri *et al.*, "Convolutional Neural Networks Applied to Neutrino Events in a Liquid Argon Time Projection Chamber," *JINST*, vol. 12, no. 03, p. P03011, 2017.

[39] C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Information Sciences*, vol. 502, pp. 279–296, 2019.

[40] Y. Suo, W. Chen, C. Claramunt, and S. Yang, "A ship trajectory prediction framework based on a recurrent neural network," *Sensors*, vol. 20, no. 18, 2020.

[41] A. Zyner, S. Worrall, and E. Nebot, "Naturalistic driver intention and path prediction using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1584–1594, 2020.

[42] G. DeZoort, S. Thais, J. Duarte, V. Razavimaleki, M. Atkinson, I. Ojalvo, M. Neubauer, and P. Elmer, "Charged particle tracking via edge-classifying interaction networks," *Computing and Software for Big Science*, vol. 5, no. 1, pp. 1–13, 2021.

[43] Y. Yao, I. Smal, I. Grigoriev, A. Akhmanova, and E. Meijering, "Deep-learning method for data association in particle tracking," *Bioinformatics*, vol. 36, pp. 4935–4941, 07 2020.

[44] A. Tsaris, D. Anderson, J. Bendavid, P. Calafiura, G. Cerati, J. Esseiva, S. Farrell, L. Gray, K. Kapoor, J. Kowalkowski, *et al.*, "The HEP.TrkX project: Deep learning for particle tracking," *Journal of Physics: Conference Series*, vol. 1085, p. 042023, sep 2018.

[45] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, p. 226–231, AAAI Press, 1996.

[46] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*, vol. 7, no. 1, pp. 48–50, 1956.

[47] M. I. Jordan, "Chapter 25 - serial order: A parallel distributed processing approach," in *Neural-Network Models of Cognition* (J. W. Donahoe and V. Packard Dorsel, eds.), vol. 121 of *Advances in Psychology*, pp. 471–495,

North-Holland, 1997.

[48] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*. USA: CRC Press, Inc., 1st ed., 1999.

[49] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[51] R. W. Schulte, V. Bashkirov, M. C. Loss Klock, T. Li, A. J. Wroe, I. Evseev, D. C. Williams, and T. Satogata, "Density resolution of proton computed tomography," *Medical Physics*, vol. 32, no. 4, pp. 1035–1046, 2005.

[52] G. Poludniowski, N. M. Allinson, and P. M. Evans, "Proton radiography and tomography with application to proton therapy.," *Br J Radiol*, vol. 88, p. 20150134, Sep 2015.

[53] R. P. Johnson, "Review of medical radiography and tomography with proton beams.," *Rep Prog Phys*, vol. 81, p. 016701, Jan 2018.

[54] H. Pettersen, J. Alme, A. Biegun, A. van den Brink, M. Chaar, D. Fehlker, I. Meric, O. Odland, T. Peitzmann, E. Rocco, *et al.*, "Proton tracking in a high-granularity digital tracking calorimeter for proton ct purposes," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 860, pp. 51–61, 2017.

[55] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, (Doha, Qatar), pp. 103–111, Association for Computational Linguistics, Oct. 2014.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.03385, 2015.

[57] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[59] T. Hastie, R. Tibshirani, and J. Friedman, "Boosting and additive trees," in *The elements of statistical learning*, pp. 337–387, Springer, 2009.

[60] R. Gluckstern, "Uncertainties in track momentum and direction, due to multiple scattering and measurement errors," *Nucl. Instrum. Meth.*, vol. 24, pp. 381–389, 1963.

[61] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, no. 3, pp. 714–735, 1997.

[62] J. F. Allen, *Natural Language Processing*, p. 1218–1222. GBR: John Wiley and Sons Ltd., 2003.

[63] K. R. Chowdhary, *Natural Language Processing*, pp. 603–649. New Delhi: Springer India, 2020.

[64] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[65] B. K. Mishra and R. Kumar, *Natural Language Processing in Artificial Intelligence*. CRC Press, 2020.

[66] I. Lauriola, A. Lavelli, and F. Aiolli, "An introduction to deep learning in natural language processing: Models, techniques, and tools," *Neurocomputing*, vol. 470, pp. 443–456, 2022.

[67] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[68] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[69] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[70] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2012.

[71] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[72] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1597–1600, 2017.

[73] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[74] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[75] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[76] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation*, vol. 31, pp. 1235–1270, 07 2019.

[77] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 324–328, 2016.

[78] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," 2017.

[79] A. Shewalkar, "Performance evaluation of deep neural networks applied to speech recognition : RNN, LSTM and GRU," *Journal of Artificial Intelligence and Soft Computing Research*, vol. Vol. 9, No. 4, pp. 235–245, 2019.

[80] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A comparison between ARIMA, LSTM, and GRU for time series forecasting," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI 2019, (New York, NY, USA), p. 49–55, Association for Computing Machinery, 2019.

[81] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[82] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example," in *2020 International*

Workshop on Electronic Communication and Artificial Intelligence (IWECAI), pp. 98–101, 2020.

[83] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[84] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015.

[85] H. Tan and M. Bansal, "LXMERT: Learning cross-modality encoder representations from transformers," 2019.

[86] H. Tang, D. Ji, C. Li, and Q. Zhou, "Dependency graph enhanced dual-transformer structure for aspect-based sentiment classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 6578–6588, Association for Computational Linguistics, July 2020.

[87] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

VII. SUPPLEMENTARY INFORMATION

Natural language processing and deep learning

Fitting reconstructed hits into a number of nodes that form an approximation to the true track trajectory can be modelled similarly to problems from the field of natural language processing (NLP). In NLP, it is common to work with sequences of words, forming sentences, and the aim is to perform tasks such as text translation, text synthesis, or speech recognition, which require algorithms that have the potential to deal with the possible different relations of entities within a sentence [62–64]. Analogously, in the problem described in this article, the reconstructed hits can be seen as an ordered (sorted along with one axis) sequence of points, which would make it straightforward for an algorithm brought from NLP to exploit those points and predict the trajectory of the track through the detector.

Nowadays, artificial intelligence (AI) is the leading choice for handling the vast majority of NLP problems, offering sophisticated algorithms that have set unprecedented results in the discipline [65, 66]. Most of these AI algorithms are categorised in the sub-field of deep learning and, more concretely, the family of "recurrent" neural networks (RNNs) [47–49] stand out. Standard feed-forward neural networks were the initial inspiration for RNNs, but RNNs highlight an extraordinary ability to learn from the semantics of temporal sequences by being trained on large amounts of data.

1. *Recurrent neural networks*

In contrast to other neural networs, RNNs can handle input sequences of different lengths and share features learnt across different positions within the sequences, mainly thanks to their capacity to use their internal states as "memory". Considering an input sequence where each position corresponds to a different time step, a standard RNN unit will produce the following activation $a^{<t>}$ and output $y^{<t>}$ for the input position x^t of the sequence at time step t:

$$\begin{aligned} a^{<t>} &= g(a^{<t-1>}, x^t; \theta_a) \\ &= g(W_{aa}a^{<t-1>} + W_{ax}x^t + b_a) \end{aligned} \tag{1}$$

$$\hat{y}^{<t>} = g(a^{<t>}; \theta_{\hat{y}}) = g(W_{\hat{y}a}a^{<t>} + b_{\hat{y}}) \tag{2}$$

where g is the activation function (e.g, hyperbolic tangent or ReLU), $a^{<t-1>}$ is the activation at time step $t-1$, and θ_a and $\theta_{\hat{y}}$ are the network parameters needed for calculating $a^{<t>}$ (i.e., W_{aa}, W_{ax}, and b_a) and $\hat{y}^{<t>}$ (i.e., $W_{\hat{y}a}$ and $b_{\hat{y}}$), respectively. Note that, for each time step t, the network is not only using the position x^t of the sequence as input but also the activation of the immediate previous time step to calculate the next activation and output. In this way, RNNs can reuse previous activations to learn about temporal information. This behaviour is depicted graphically in Fig. 8. It is relevant to mention that the network parameters are shared over time, meaning that the model size does not increase with the length of the input sequence.

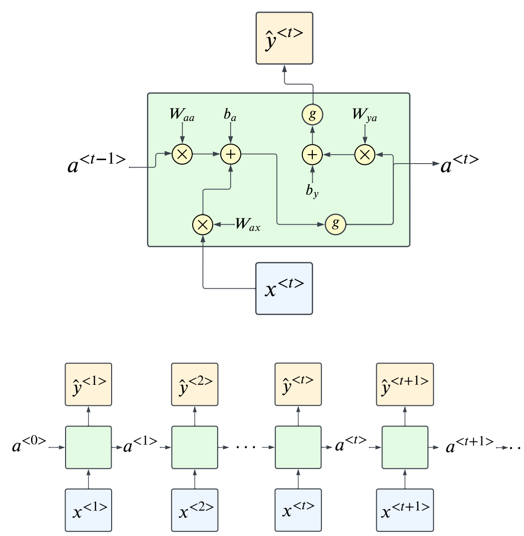

FIG. 8. (Top) Internal structure of a recurrent neural network (RNN) unit for the time step t, where the input position x^t of the sequence and the previous activation $a^{<t-1>}$ are used to calculate the next activation $a^{<t>}$ and output $\hat{y}^{<t>}$; (bottom) unfolded structure of a standard RNN, where the activation at one time step becomes an input to the next time step.

Having the output of the network $\hat{y}$ and the true labels y, the discrepancy between the two is evaluated with the following loss function $\mathcal{L}$:

$$L(\hat{y}, y) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\hat{y}, y) \tag{3}$$

where T is the total number of time steps. In RNNs, the model weights θ are updated during backward propagation at each time step, what is generally called back-propagation through time:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \mathcal{L}(\hat{y}_t, y_t)}{\partial \theta} \tag{4}$$

In the above scenario, in order to make predictions on the current position, the model can learn about the previous part of the sequence. However, all the following positions are ignored. In other words, the model has the ability to learn from the "past" but not from the "future". Accessing future information might be necessary to report accurate results in some cases. For example, coming back to the physics problem presented in this manuscript, to precisely predict the closest 3D position to the actual particle trajectory for a particular reconstructed hit, it might be advantageous to access both the previous and the next hits within the sequence. The solution to also learn about the future is to put two independent RNNs together into what is called a bidirectional recurrent neural network (BRNN) [67], where the input is given from start to end to one RNN and from back to the front to the other RNN; then, the outputs at each time step usually are concatenated, as illustrated in Fig. 9. In this way, for each time step, the network has access to the activations coming from the previous position and the following position in the sequence, giving the model the ability to learn from the past and the future simultaneously.

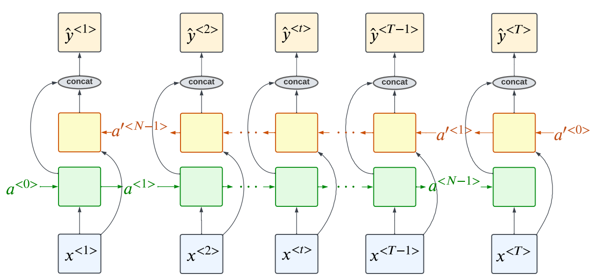

FIG. 9. Overview of a standard bidirectional recurrent neural network (BRNN) architecture. The architecture consists of two RNNs combined together. The input sequence is given from start to end (from left to right in the figure, in green) to one of them, and from back to front (right to left, in yellow) to the other one. The output of each RNN is normally concatenated for each time step.

Figures 8 and 9 show the case where the length of the input sequence matches the length of the output; one example of this could be a problem where the goal is to categorise each word in a sentence into the corresponding category (e.g., noun, pronoun, verb, or adjective). Another example, which is solved in this manuscript, is to predict the closest track trajectory point for each input reconstructed hit. Nevertheless, there are many other RNN topologies: many-to-one, where only the output of the last time step is considered (e.g., for sentiment classification); or many-to-many, but, in this case, the length of the output sequence does not necessarily have to match the length of the input sequence (e.g., text translation or music generation).

2. *GRU and LSTM*

Some sequence models might be affected by very long-term dependencies, meaning that, within a sequence, it could be possible to find strong relations such as the dependency of an arbitrary position i and a position $i + k$, being k a large positive integer. On top of that, since the input sequences can have different lengths, the long-term dependencies can be arbitrarily long.

Due to the continuous recalculation of the activations (shown in Equation 1), standard RNNs are not good at catching long-term dependencies, arising vanishing/exploding gradient problems during back-propagation [68–70]. Several architectures have been proposed to deal with this issue, where gated recurrent and long short-term memory units stand out.

A gated recurrent unit (GRU) [55, 71–73] is an alternative to the original RNN approach that handles long-term dependencies by calculating a candidate $\tilde{a}^{<t>}$ of the activation (Eq. 5) using a gate to measure how relevant the previous activation is to compute the next candidate (Eq. 6).

$$\begin{aligned}\tilde{a}^{<t>} &= tanh(a^{<t-1>}, x^t; \theta_a) \\ &= tanh(W_{aa}(\Gamma_r \odot a^{<t-1>}) + W_{ax}x^t + b_a)\end{aligned} \tag{5}$$

$$\Gamma_r = \sigma(W_{ra}a^{<t-1>} + W_{rx}x^t + b_r) \tag{6}$$

where *tanh* and *sigma* are the hyperbolic tangent function and the sigmoid function, respectively. The activation is then updated using another gate (Eq. 7) to weight the candidate and the previous activation into the new activation (Eq. 8). Figure 10 represents the GRU workflow as a whole.

$$\Gamma_u = \sigma(W_{ua}a^{<t-1>} + W_{ux}x^t + b_u) \tag{7}$$

$$a^{<t>} = \Gamma_u \odot \tilde{a}^{<t>} + (1 - \Gamma_u) \odot a^{<t-1>} \tag{8}$$

Similarly to GRU, the long short-term memory (LSTM) [74–76] unit handles long-term dependencies by not only updating the activation $a^{<t>}$ at each time step but also updating a new entity named the "memory" cell $c^{<t>}$. The LSTM unit uses three different gates: (1) a gate Γ_u (Eq. 7, equivalent to th GRU version) that tells how much the memory cell candidate $\tilde{c}^{<t>}$ (Eq. 9) should affect the update of the new memory cell $c^{<t>}$ (Eq. 10); (2) a gate Γ_f (Eq. 11) that measures how much to forget about the previous memory cell $c^{<t-1>}$ during the new

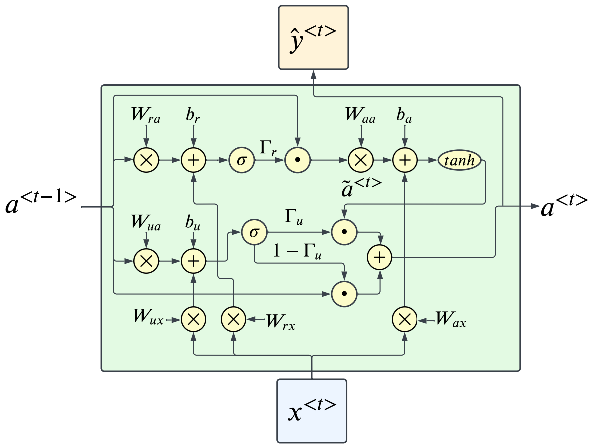

FIG. 10. Gated recurrent unit (GRU) structure. For each time step, the input $x^{<t>}$ and the previous activation $a^{<t-1>}$ are used to compute the new activation $a^{<t>}$ and the output $\hat{y}^{<t>}$. To do so, the gate Γ_r is used to give relevance to the previous activation when calculating the activation candidate $\tilde{a}^{<t>}$, while the gate Γ_u is used to update the activation by weighting the candidate $\tilde{a}^{<t>}$ and the previous activation $a^{<t-1>}$.

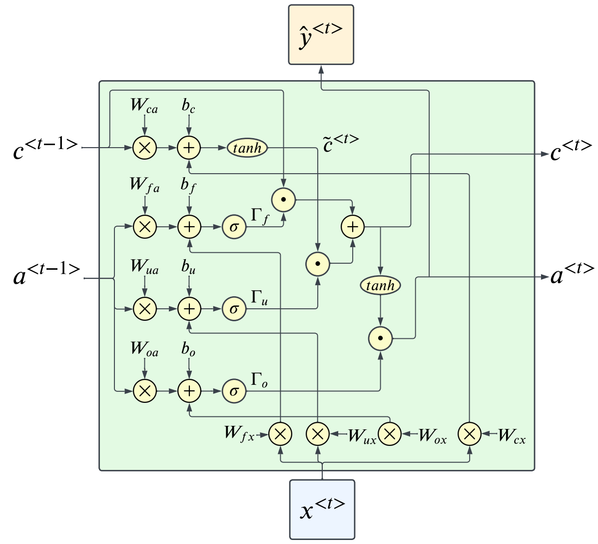

FIG. 11. Long short-term memory (LSTM) unit. For each time step, the input $x^{<t>}$, and the previous activation $a^{<t-1>}$ and memory cell $c^{<t-1>}$ are used to compute the new activation $a^{<t>}$, memory cell $c^{<t>}$, and output $\hat{y}^{<t>}$. The gates Γ_u and Γ_f contribute to the computation of the new memory cell $c^{<t>}$, while the gate Γ_o is used to calculate the activation $a^{<t>}$.

memory cell $c^{<t>}$ calculation; and (3) a gate Γ_o (Eq. 12) that weights the memory cell during the calculation of the new activation $a^{<t>}$ (Eq. 13). Figure 11 helps understand the above formulas for the LSTM unit by showing the different calculations in a diagram.

$$\begin{aligned}\tilde{c}^{<t>} &= tanh(c^{<t-1>}, x^t; \theta_c)\\ &= tanh(W_{cc}(\Gamma_r \odot c^{<t-1>}) + W_{cx}x^t + b_c)\end{aligned} \tag{9}$$

$$c^{<t>} = \Gamma_u \odot \tilde{c}^{<t>} + \Gamma_f \odot c^{<t-1>} \tag{10}$$

$$\Gamma_f = \sigma(W_{fa}a^{<t-1>} + W_{fx}x^t + b_f) \tag{11}$$

$$\Gamma_o = \sigma(W_{oa}a^{<t-1>} + W_{ox}x^t + b_o) \tag{12}$$

$$a^{<t>} = \Gamma_o \odot tanh(c^{<t>}) \tag{13}$$

In practice, both GRU and LSTM perform similarly in terms of the quality of the results for different problems [77–80]. However, due to the lack of a memory unit and thus requiring fewer calculations, GRU tends to be the preferred choice over LSTM since the former is more computationally efficient [81, 82].

3. *Transformers*

Even though RNNs and their variants GRU and LSTM have reported remarkable results in the field of NLP, they still face a couple of drawbacks. On the one hand, sequences are processed position by position and not altogether, with the risk of still forgetting information regardless of the memory mechanisms, which might not retain all the necessary relations among positions within the sequence. On the other hand, in order to learn about the "past" and the "future" for each time step, bidirectional models are needed, which require twice the usual computation.

A Transformer [50] is a type of neural network, initially proposed for text translation but with many different current applications, that resolves the issues above by treating each input sequence as a whole. Its main feature is the multi-head self-attention mechanism (revolutionising the attention proposed in [83] and [84]), which decides the fragments of the input sequence that are more relevant for the target task by capturing correlations among all items in a sequence. Formally, an input sequence $X \in R^{N \times d_k}$ (sequence of length N, each position represented by d_k values) is multiplied by three weight matrices W_Q, W_K, and $W_V \in R^{d_k \times d_{model}}$ (where d_{model} the length of the new representation for each position of the sequence after the attention mechanism) to produce Q (queries), K (keys), and V (values) $\in R^{N \times d_{model}}$, respectively. The self-attention is calculated with the following formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{14}$$

The multi-head part implies repeating the self-attention h times (each self-attention calculation with independent learnt parameters is known as a "head") on Q, K, and V projected through h sets of weight matrices $W^Q_{(i)}$, $W^K_{(i)}$, and $W^V_{(i)} \in R^{d_{model} \times d_k}$ (in the multi-head approach, $d_k = d_{model}/h$.). Then, the outputs of the heads are concatenated and multiplied by a final weight matrix $W^O \in R^{hd_k \times d_{model}}$, as shown in Eqs. 15 and 16 and Fig. 12:

$$Multi\text{-}head(Q, K, V) = concat(head_{(1)}, ..., head_{(h)})W^O \tag{15}$$

where:

$$head_{(i)} = Attention(QW^Q_{(i)}, KW^K_{(i)}, VW^V_{(i)}) \tag{16}$$

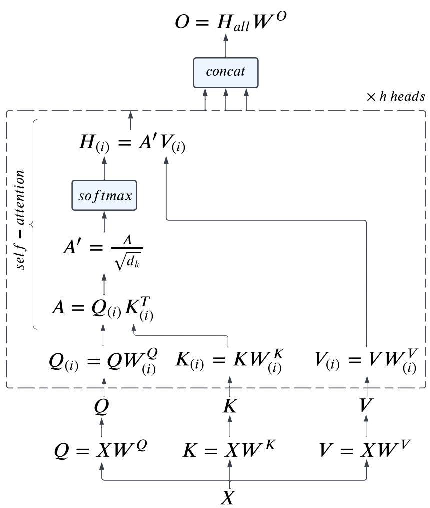

FIG. 12. Multi-head attention. The input X is linearly projected into Q, K, and V (the initial W^Q, W^K, and W^V matrices are usually the same, resulting in equivalent Q, K, and W; in NLP problems, this first linear projection is called "embedding"), which are processed h times (one per head) through a self-attention mechanism. The output of the heads are concatenated and multiplied by a final weight matrix W^O to produce the output O.

In order to preserve the order of the input sequence through the different projections and let the model learn about relative positions, "positional encodings" are summed to the first linear projection of the input. The authors of the Transformer model chose sine and cosine functions of different frequencies for the positional encoding [50]:

$$PE_{(pos,i)} = \begin{cases} sin(\frac{pos}{10000^{i/d_{model}}}) & \text{if } i \text{ is even} \\ cos(\frac{pos}{10000^{(i-1)/d_{model}}}) & \text{if } i \text{ is odd} \end{cases} \tag{17}$$

where pos is the position in the sequence and i is the dimension.

Before putting everything together, it is advised to mention that the original Transformer architecture consists of two main components: an encoder and a decoder. In machine translation, the encoder learns relevant features from the input sequence that are useful for the decoder to generate the translated sequence sequentially. In the Transformer model, the inputs are first projected through a linear layer in addition to applying the positional encoding to finally go through the encoder, which consists of a multi-head attention module, an addition (of the output and the input of the multi-head attention) and a normalisation, followed by another linear layer and a final addition+normalisation, all repeated N times. Similarly, the outputs (shifted right) are projected through a linear layer. A positional encoding is applied, to then go through the decoder, which consists of a multi-head attention module, an addition+normalisation, another multi-head attention module (where the input queries Q and keys K are the output of the encoder, which lets the decoder decide which encoder input is relevant for the decoding task), another addition+normalisation, a linear layer, and a final addition+normalisation, all repeated N times. The procedure described is depicted in Fig. 13.

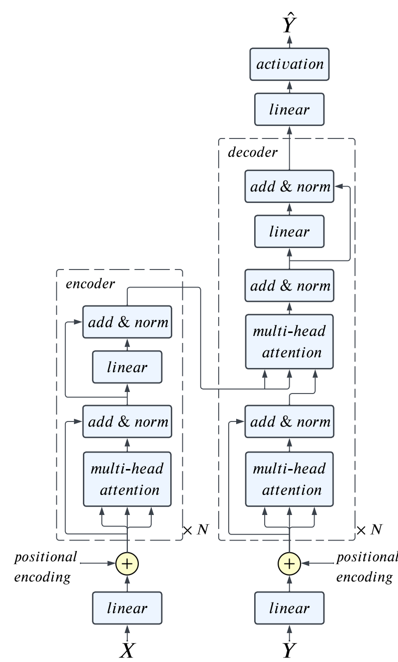

FIG. 13. Transformer encoder-decoder architecture. Figure based on the model published in [50].

The original architecture needs the decoder part since it was designed for machine translation. However, for other problems, such as sentiment analysis, the decoder can be replaced with a simpler module (e.g., a linear layer) [85–87] since there is no need to predict an output sequence but, for example, a single label. In other cases, like the model proposed in this article, the goal is to predict an output for each item in the input sequence; thus, the decoder can be omitted and substituted by a linear layer.