Eur. Phys. J. C (2024) 84:691
https://doi.org/10.1140/epjc/s10052-024-13047-3

THE EUROPEAN PHYSICAL JOURNAL C

Regular Article - Experimental Physics

Machine-learning-based particle identification with missing data

Miłosz Kasak[1], **Kamil Deja**[2,1,a], **Maja Karwowska**[1,3], **Monika Jakubowska**[1], **Łukasz Graczykowski**[1], **Małgorzata Janik**[1]

[1] Warsaw University of Technology, pl. Politechniki 1, 00-661 Warsaw, Poland
[2] IDEAS NCBR, Chmielna 69, 00-801 Warsaw, Poland
[3] CERN-European Organization for Nuclear Research, Espl. des Particules 1, 1211 Geneva, Switzerland

Received: 12 December 2023 / Accepted: 17 June 2024

Abstract In this work, we introduce a novel method for Particle Identification (PID) within the scope of the ALICE experiment at the Large Hadron Collider at CERN. Identifying products of ultrarelativisitc collisions delivered by the LHC is one of the crucial objectives of ALICE. Typically employed PID methods rely on hand-crafted selections, which compare experimental data to theoretical simulations. To improve the performance of the baseline methods, novel approaches use machine learning models that learn the proper assignment in a classification task. However, because of the various detection techniques used by different subdetectors, as well as the limited detector efficiency and acceptance, produced particles do not always yield signals in all of the ALICE components. This results in data with missing values. Out of the box machine learning solutions cannot be trained with such examples without either modifying the training dataset or re-designing the model architecture. In this work, we propose the new method for PID that addresses these issues and can be trained with all of the available data examples, including incomplete ones. Our approach improves the PID purity and efficiency of the selected sample for all investigated particle species.

1 Introduction

ALICE (A Large Ion Collider Experiment) [1] is one of the four major detectors located at the Large Hadron Collider at CERN [2]. The main goal of ALICE is to study the properties of quark–gluon plasma (QGP), a hot and dense state of matter, and the strong force that binds quarks together inside hadrons [3]. The key requirement for detailed studies of QGP that distinguishes ALICE from the other Large Hadron Collider (LHC) experiments is its capability for very precise particle identification (PID) – i.e. the ability to discriminate between different particle species produced during the collision. This allows for selecting a subset of particles required for specific analysis.

The ALICE experiment is composed of several subdetectors, some of which measure particle properties that can be used for identification. Figure 1 presents a scheme of the detector in Run 2, the previous LHC data-taking periods.

The three detectors particularly useful for PID are: TPC, TOF, and TRD. The Time Projection Chamber [5] (TPC) is one of the most important ALICE detectors as it records 3D information of the trajectory of charged particles, as well as their specific energy loss due to ionization, which is essential for particle identification. The Time-of-Flight [6] (TOF) detector measures particle travel times from the collision vertex to the detector, from which the particle velocity and mass are calculated. The Transition Radiation Detector [7] (TRD) records transition radiation, that is, the emission of photons by electrons traversing the boundaries of a radiator, which helps in distinguishing electrons from other charged particles. All the detectors mentioned above detect particles carrying a non-zero electric charge. Therefore, this article focuses on the identification of charged particles.

With the signals recorded by the detectors described above, particles are chosen using a set of selection criteria. Traditionally, the particle identification is based on hand-crafted selection criteria, for instance, based on how much the detector response deviates from the expected value. Particles falling outside the selection region are rejected. One of the most common selection methods is described in detail in Sect. 2.3.

However, when the characteristics of different particle species overlap, combining information from multiple detectors becomes difficult. Choosing selection regions by trial and error is less effective in this case, lowering the purity of the selected sample.

[a] e-mail: kamil.deja@pw.edu.pl (corresponding author)

Published online: 13 July 2024

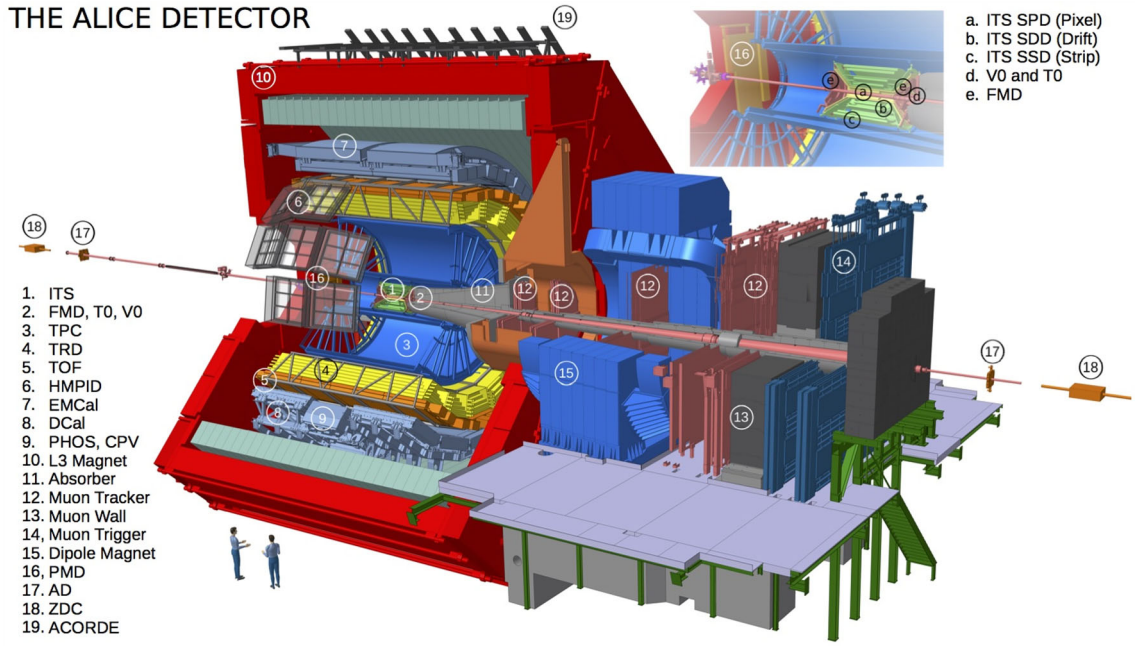

Fig. 1 Components of the ALICE detector in its Run 2 configuration [4]

These shortcomings can be addressed with more advanced classification methods such as Bayesian models [8] or neural-network-based (NN) approaches [9–11]. However, the more sophisticated methods can only be trained or employed with complete data instances; just as a function value cannot be computed without all the inputs, NNs may not be used to process examples with missing values. As detectors used for particle identification work independently, particles may be measured by a subset of detectors while not being recorded by others. This can happen for various reasons; for instance, a detector may malfunction or be switched off, or the particle can have characteristics (e.g. too low transverse momentum) that do not match the detector specification. Collected data may therefore contain incomplete measurements, with information from only some detectors. As a result, standard neural-network-based approaches cannot be used for PID without altering either the incomplete data or the model architecture. The former approach is usually implemented by inputting average or predicted missing values. This introduces artificial information to the data that can change the predictions of a model in undefined ways. Such an approach can often lead to biased outputs, which is especially problematic for real-life applications such as PID, where model predictions may lead to physically impossible outcomes. Therefore, we focus on altering the model architecture, as this approach allows us to avoid making any assumptions about the missing data and ensures that no contained information is modified.

In this work, we introduce the new method of PID with missing data. In particular, we propose an adaptation of an attention-based multi-instance learning approach [12]. The resulting model can predict and learn from examples containing any number of missing measurements and allows for using knowledge learned from complete examples when classifying incomplete ones, as well as the other way around.

Our tests show that this additional knowledge improves the performance of neural-network-based models. We evaluate different methods for coping with incomplete data and show that our approach achieves state-of-the-art performance in PID with neural networks. We attribute this performance gain to the proposed method that learns from all available data, including examples with missing data.

2 Related works

2.1 Missing data sources

The incomplete data problem in statistical and machine learning (ML) models has been widely studied over decades [13, 14]. In [15], authors highlight three main categories of missing data according to the source of the information gaps. The first case occurs when examples are *missing completely at random* (MCAR) – where the probability of missing value is independent of all parameters. If the probability of the missing value is dependent only on the value of the observed attributes, we can call it *missing at random* (MAR). Finally, if this probability depends on the value of the missing parameter itself, we refer to this case as *missing not at random* (MNAR).

In our case of the data used for particle identification at the ALICE experiment, we can distinguish two main sources of the missing data. For some cases, the lack of particular value – signal from one of the detectors is caused by its temporal malfunction, independent of the measured particle. In this case the signal is missing completely at random (MCAR). On the other hand, for some cases we might lack the measurement of values that fall outside the effective region of particular detector. For example, the efficiency of the TOF detector measurements drop significantly for particles with transverse momentum below 0.5 GeV/c. In such case, the missing data is missing not at random (MNAR).

2.2 Machine learning with incomplete data

Several popular machine learning algorithms such as neural networks, cannot be directly trained with missing data. Therefeore there exists different solutions tot that problem that can be roughly divided into two groups: (1) those that modify the training dataset [16,17] and (2) those that adapt ML architectures [18–21]. Data treatment methods transform datasets with missing values into complete ones that can be used with standard machine learning architectures. Adapted architectures are capable of processing incomplete data without any alteration.

The simplest method for data transformation is known as *case deletion*, where examples without full data availability are simply removed from the training set. This limits the potential training capabilities of the model and can result in bias induced by missing part of the original data distribution. The alternative technique known as *imputation* fills the missing data with artificial values calculated as a *mean*, *median*, or value predicted with a simple model, e.g. *linear regression*. This idea is further extended in [22], where the imputation model is optimized jointly with the regressor. Similarly to the case deletion approach, imputation methods can also significantly disturb the predictions of the neural model, which might result in unrealistic behavior.

To overcome those shortcomings, methods based on the model's adjustment were proposed. The most straightforward approach suitable in a situation when missing data affects only a few attributes is an ensemble of different classifiers. In such a case, different models are trained on subsets of the training dataset without missing data. In particular, in *neural network reduction* [21], authors propose to split the dataset into the largest possible complete subsets.

There are several domains where machine learning with incomplete data is already employed. In [23], authors use a recurrent neural network to model healthcare data with missing data. A similar idea was employed in medical applications such as breast cancer prediction [24]. Machine learning algorithms with missing data are also used in other domains such as traffic prediction [25,26].

2.3 Particle identification techniques

As mentioned in the introduction, the recently employed PID techniques depend on the human-defined selection criteria on the response signal from given detectors used in the analysis. In most cases, the so-called "n_σ" method is used, whose main parameter is the number of standard deviations from the expected value that a signal for a given particle left in the detector. For example, if both the TPC and TOF detectors are used, a common approach is to define a PID selection as

$$\sqrt{n^2_{\sigma,\mathrm{TPC}} + n^2_{\sigma,\mathrm{TOF}}} < \lambda. \tag{1}$$

The actual cut-off value λ, however, depends on the specific analysis, and requirements for purity and efficiency of the sample. Typically, this value is set between 2 and 3. This method can be further modified by adding additional conditions, for instance, on rejection of other types of particles. In such a case, if one analyzes, i.e., charged kaons, one may provide the acceptance criterion for the kaon hypothesis (i.e. how far in terms of n_σ the signal can be from the expected value for kaons) as well as the rejection criterion for the pion, proton, and electron hypothesis (i.e. what is the minimum n_σ that the signal must have compared to the expected values for pions, protons, and electrons).

Outside of the ALICE experiment, several attempts have been made to use neural networks for the PID. In the LHCb experiment [9], shallow neural networks are used to classify calorimeter signals. Similarly, in ATLAS, neural models are used to identify electrons [10], while the CMS Collaboration identifies τ lepton decays with deep neural networks [11].

3 Attention-based neural network architecture for particle identification

In this work, we focus on the problem of PID in which a type of particle is assigned to a data sample based on its characteristics recorded by a set of detectors. We treat particle identification as a set of binary classification tasks in a so-called *one vs. all* approach. Given a set of examples, each labeled with one of k particle types, the task is to find k binary classifiers corresponding to each particle type. Each classifier is tasked with identifying whether or not a particle, represented by a given example, belongs to the specific particle type. Every example consists of a set of real-valued features, based on measurements from a detector, out of which some might be missing.

With this formulation, we introduce a novel method for PID that employs missing data. We base our approach on the attention mechanism, similar to the method introduced for a medical use-case in AMI-Net [12]. Our system is com-

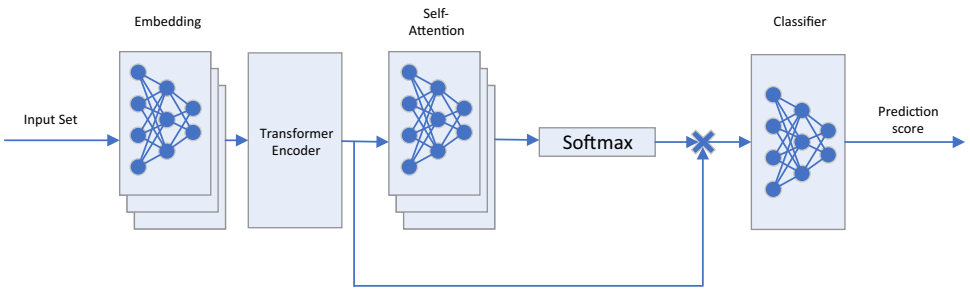

Fig. 2 The proposed model architecture. Layered blocks are applied separately to each vector in a set. Single blocks are applied to their input as a whole

posed of several steps. We first encode all data examples into a set of feature-value pairs [27], independently of their characteristics regarding missing data, and transform them into embeddings. These embeddings are further processed by the Transformer's [28] encoder module with multi-head attention. The encoder output is combined into a single feature vector that is an input to the final classifier. The overview of our system is shown in Fig. 2, while in the following subsections, we describe its building blocks in detail.

3.1 Feature value pairs

We prepare the incomplete data for the model by creating sets of feature-value pairs from the incomplete vectors. Each pair consists of a non-missing value in the vectors and the index of that value. We apply one-hot encoding to these indices to aid the model in their processing. An example is shown in Table 1.

3.2 Embedding

Embedding is a continuous vector representation of discrete data. Embedded feature indices can use relations between features more effectively by placing similar features close in the embedded space. We create embeddings by applying a neural network with a single hidden layer to the feature-value vectors.

3.3 Transformer encoder

Transformer is an attention-based architecture commonly used for sequence transduction. As the Transformer is used to process sequences of arbitrary lengths, it can naturally be adapted to find/learn relations between available features regardless of the amount of missing values. We apply the Transformer's encoding module to the set of embedding vectors to connect the different features each vector represents.

The encoder consists of a stack of N identical layers. Each layer is made of two sub-layers: a multi-head attention layer and a dense neural network. The multi-head attention is applied to the input set as a whole, while the NN is applied to each vector in a set separately. To ease the training of the encoder, we apply a residual connection around each sub-layer, followed by layer normalization.

3.3.1 *Multi-head attention*

To connect pairs of vectors in the input set, each input vector is linearly transformed into h query, key, and value vectors, and scaled dot-product attention is applied to each of the h triples of query, key, value sets, where h is the number of heads. This allows specific patterns to be found in pairs of input vectors. For example, a measurement from a specific detector could be used if the momentum is within a particular range. As each of the N layers in the encoder contains a multi-head attention sub-layer, and each sub-layer connects pairs of vectors, the full encoder can theoretically connect subsets of 2^N vectors.

3.3.2 *Scaled dot-product attention*

Given sets of query, key, and value vectors of dimension d_k, scaled dot-product attention calculates a set of weighted averages of the value vectors based on the similarities between the corresponding keys and queries. The similarities are obtained by computing the dot product of the query and key vectors. The dot products are scaled by a factor of $\frac{1}{\sqrt{d_k}}$ to counteract the increase of the dot product with an increasing dimension of the query, key, and value vectors. Finally, the softmax function is applied to the scaled dot products to obtain the weights used to calculate the averages of value vectors. The whole computation can be described as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2}$$

where $Q, K, V \in \mathbb{R}^{n\times d_k}$ are sets of n query, key and value vectors, respectively.

3.4 Attention pooling

As the classifier neural network cannot process an unordered, variable-size set of vectors, merging the vectors obtained from the Transformer's encoder into a single vector is necessary. Therefore, we pool the output vectors together by

Table 1 An example for preprocessing of data samples into feature set values

(a) 3 data samples with 4 attributes with different amounts of missing values.

id	momentum	TOF	TPC	TRD
1	0.1		3	
2	7	70	24	13
3		78		

(b) First particle.

key				value
1	0	0	0	0.1
0	0	1	0	3

(c) Second particle.

key				value
1	0	0	0	7
0	1	0	0	70
0	0	1	0	24
0	0	0	1	13

(d) Third particle.

key				value
0	1	0	0	78

using the self-attention technique that was also used in the Transformer, where each vector is assigned weights based on its own values. These weights are then used to calculate the weighted average of all the vectors in the set.

3.4.1 Self attention

Given a set of vectors $\{v_1, v_2, ..., v_n\}$, where $v_i \in \mathbb{R}^{d_{model}}$, we obtain the pooled vector as follows:

$$E_{i,\star} = \mathrm{NN}(v_i) \quad \forall i \quad \in [1, n] \tag{3}$$

$$A_{\star,j} = \mathrm{softmax}(E_{\star,j}) \quad \forall j \quad \in [1, d_{model}] \tag{4}$$

$$o_j = \sum_{k=1}^{n} A_{k,j} v_{k_j} \quad \forall j \quad \in [1, d_{model}] \tag{5}$$

Here, $E, A \in \mathbb{R}^{n \times d_{model}}$ are respectively the self-attention values and weights, $o \in \mathbb{R}^{d_{model}}$ is the pooled output vector, and NN is a neural network that has both dimensions of input and output equal to d_{model}.

3.5 Classifier

We obtain the final prediction score for the PID task by applying a simple neural network with a single-value output to the pooled vector. We normalize the score to the range (0, 1) by applying the logistic function:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{6}$$

The obtained score approximates the probability of the example corresponding to a specific particle type.

4 Evaluation

We evaluate our model on the PID task by comparing it against a standard n_σ-based selection technique, an ensemble of neural networks [21] as well as two standard techniques for ML with missing data: mean imputation, and linear regression imputation. Additionally, in a scenario with no missing data in the test-set, we compare our approach to the case deletion procedure.

4.1 Dataset

The data comes from a Monte Carlo simulation of proton–proton collisions at $\sqrt{s} = 13$ TeV with a realistic simulation of the time evolution of the detector conditions in the LHC Run 2 data-taking period. The simulation was performed with PYTHIA 8 [29], the GEANT [30] particle transport model, and general-purpose settings.

The dataset consists of 2,751,934 examples with track transverse momentum $p_\mathrm{T} \geq 0.1$ GeV/c. 95% of examples fall into the range [0.12, 1.76] GeV/c. Each example contains 19 features:

- detector signals: 1 TPC feature, 2 features per TRD and TOF
- number of shared TPC clusters
- spatial coordinates of a track reconstruction starting point (x, y, z in the local coordinate system, and the rotation angle α between local and global coordinate systems),
- track momentum p and its p_T, p_x, p_y, p_z components,
- track charge and type (propagated/non-propagated Run 3 track, Run 2 track)
- the distances of closest approach (DCA) of the track trajectory to the collision primary vertex measured in the xy plane (d_{xy}) and the z direction (d_z)

Table 2 Particle type distribution. Approximately 97.8% of the examples belong to the 6 most populous particle types

Pion	Kaon	Proton	Electron	Muon
43.59%	3.415%	2.026%	0.794%	0.323%
Antipion	Antikaon	Antiproton	Antielectron	Antimuon
43.66%	3.288%	1.819%	0.762%	0.321%

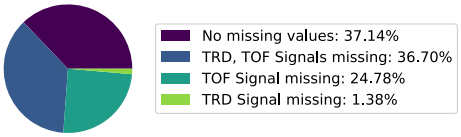

Fig. 3 Missing data distribution. Over 62.8% of the examples are missing at least one value

The examples are labeled with ten different particle types. The particle species distribution is shown in Table 2.

There are four combinations of missing values, as some examples' measurements from the TOF and TRD detectors are missing. Figure 3 shows the missing value distribution.

4.2 Models

For each compared method, we train identical binary classification models for 6 of the most populous particle types: pions, protons, kaons, and their respective antiparticles.

All trained models are identical for imputation methods and the neural network ensemble. They have three hidden layers of sizes 64, 32 and 16, and a single output. Between the layers, we use Rectified Linear Unit (ReLU) activation function $f(x) = \max(0, x)$. Dropout regularization with a rate of 0.1 is applied after each activation layer. The input dimension depends on the method used. For imputation techniques, all models process inputs of size 19, as all missing features are imputed. For the neural net ensemble, the four networks have input sizes 19, 17, 17, and 15, depending on the combination of missing values from the TRD and TOF detectors (each detector has two features associated with it).

Our proposed architecture consists of the embedding neural network, the Transformer's encoder, the self-attention neural network, and the classifier network. ReLU activation is also used between neural network layers, and dropout regularization with a rate of 0.1 is applied to the output of the embedding layer and the output of each sub-layer in the encoder. The parameters of all the layers are shown in Table 3.

Model parameters are selected using a hyperparameter sweep for a fair comparison of different network architectures. Given a set of parameter, various combinations of their values are used to train different models, from which the model achieving the best results on the validation dataset is chosen. In this way, each of the compared methods may achieve the best possible results. The hyperparameter sweep is a computationally expensive procedure, hence we perform it with a model trained to detect kaons – the most challenging particles to identify in our dataset. This might bring a small bias of our results towards kaons, so before the full integration of our solution with the ALICE system, we will perform independent sweeps for all particles.

4.3 Results

Each model is trained using all complete and incomplete examples and tested in two cases: (1) with all available test examples and (2) with the complete ones only. For case deletion, no results are available for testing on the missing data dataset, since it is not possible to directly apply the appropriate models on data with missing values. We evaluate our models on test-set with standard metrics – precision (purity) and recall (efficiency). Since precision and recall are antagonistic, we also include the F_1 metric, which is a combination of the first two according to the formula:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{7}$$

Numerical results for the pion, proton, and kaon particles are shown in Tables 4 and in 5 for antiparticles. We highlight the best-performing method for each particle in bold.

For the case of incomplete examples, we compare our machine learning solutions with the standard technique described in Sect. 2.3. For our comparison we used the following selections: $|n_{\sigma,\mathrm{TPC}}| < 3$ for particles with transverse momenta below 0.5 GeV/c and $\sqrt{n^2_{\sigma,\mathrm{TPC}} + n^2_{\sigma,\mathrm{TOF}}} < 3$ for particles with $p_{\mathrm{T}} \geq 0.5$ GeV/c (in this case TOF signal was required). Tables 4 and 5 show that machine learning approaches, in general, outperform standard n_σ-based techniques, providing significantly higher recall for similar or higher precision. Moreover, the proposed architecture is comparable with other tested techniques. We can also observe that training with additional examples with missing data still results in the good quality of the PID on complete examples, as measured by F_1 scores. On the other hand, synthetic imputation of mean or predicted missing values can disturb the learned function and might result in lower performance. The cases of pion and kaon identification on complete cases are the only two where the ensemble achieves slightly better F_1 results than the proposed architecture.

4.4 Detailed analysis of the results

To further highlight the benefits of our approach, we provide a detailed analysis of its performance when compared to the baseline approaches. In Fig. 4, we present precision-recall curves of different methods applied on all available test

Table 3 Training hyperparameters

Embedding			Transformer encoder						Self-attention			Classifier		
			Encoder layer					Num layers						
			Multi-head attention		Neural network									
Input	Hidden	Output	Dimension	Heads	Input	Hidden	Output		Input	Hidden	Output	Input	Hidden	Output
20	128	32	32	2	32	128	32	2	32	64	32	32	64	1

Table 4 Classification result for the three most common particle species

Model	Precision	Recall	F_1
(a) Pion identification on complete data only			
Delete	99.08 ± 0.07	99.67 ± 0.04	99.37 ± 0.01
Ensemble	**99.11 ± 0.04**	99.64 ± 0.06	**99.38 ± 0.01**
Mean	98.85 ± 0.09	**99.69 ± 0.04**	99.27 ± 0.04
Proposed	99.08 ± 0.02	99.64 ± 0.03	99.36 ± 0.01
Regression	99.02 ± 0.02	99.49 ± 0.14	99.25 ± 0.07
(b) Proton identification on complete data only			
Delete	99.23 ± 0.32	99.63 ± 0.05	99.43 ± 0.16
Ensemble	99.16 ± 0.24	**99.76 ± 0.07**	99.46 ± 0.13
Mean	99.22 ± 0.19	99.72 ± 0.04	99.47 ± 0.08
Proposed	**99.28 ± 0.10**	99.68 ± 0.09	**99.48 ± 0.02**
Regression	99.10 ± 0.09	99.65 ± 0.09	99.37 ± 0.07
(c) Kaon identification on complete data only			
Delete	**96.93 ± 0.37**	96.98 ± 0.26	96.95 ± 0.06
Ensemble	96.65 ± 0.38	97.82 ± 0.31	**97.23 ± 0.10**
Mean	96.83 ± 0.17	95.33 ± 0.67	96.08 ± 0.36
Proposed	96.03 ± 0.98	**98.06 ± 0.72**	97.04 ± 0.17
Regression	94.27 ± 0.98	97.01 ± 0.51	95.62 ± 0.39
(d) Pion identification on data including incomplete examples			
Standard	**99.99 ± 0.01**	78.37 ± 0.01	87.87 ± 0.87
Ensemble	97.47 ± 0.25	99.46 ± 0.21	98.45 ± 0.04
Mean	97.31 ± 0.07	99.52 ± 0.07	98.40 ± 0.01
Proposed	97.49 ± 0.06	**99.54 ± 0.05**	**98.50 ± 0.02**
Regression	97.33 ± 0.06	99.49 ± 0.07	98.40 ± 0.04
(e) Proton identification on data including incomplete examples			
Standard	**99.40 ± 0.01**	59.72 ± 0.03	74.61 ± 1.88
Ensemble	97.16 ± 0.46	93.74 ± 0.30	95.42 ± 0.12
Mean	97.85 ± 0.41	93.34 ± 0.32	95.54 ± 0.06
Proposed	97.80 ± 0.44	**93.86 ± 0.27**	**95.79 ± 0.07**
Regression	97.38 ± 0.40	93.67 ± 0.38	95.49 ± 0.15
(f) Kaon identification on data including incomplete examples			
Standard	**92.87 ± 0.01**	60.37 ± 0.05	73.17 ± 1.57
Ensemble	91.18 ± 02.00	82.72 ± 01.42	86.74 ± 0.16
Mean	90.83 ± 01.71	82.32 ± 0.96	86.36 ± 0.34
Proposed	91.55 ± 0.71	**83.68 ± 0.82**	**87.44 ± 0.14**
Regression	91.17 ± 01.00	81.78 ± 0.21	86.22 ± 0.46

Table 5 Classification result for the three most common antiparticle species

Model	Precision	Recall	F_1
(a) Antipion identification on complete data only			
Delete	**99.08 ± 0.04**	99.67 ± 0.02	**99.37 ± 0.01**
Ensemble	98.93 ± 0.51	**99.76 ± 0.16**	99.34 ± 0.18
Mean	98.86 ± 0.11	99.69 ± 0.03	99.27 ± 0.04
Proposed	**99.08 ± 0.04**	99.67 ± 0.02	**99.37 ± 0.03**
Regression	99.04 ± 0.03	99.51 ± 0.05	99.28 ± 0.02
(b) Antiproton identification on complete data only			
Delete	98.75 ± 0.37	99.52 ± 0.17	99.13 ± 0.26
Ensemble	99.12 ± 0.08	99.53 ± 0.16	99.33 ± 0.10
Mean	98.79 ± 0.58	99.62 ± 0.18	99.20 ± 0.27
Proposed	**99.25 ± 0.06**	99.63 ± 0.20	**99.44 ± 0.08**
Regression	98.57 ± 0.33	**99.64 ± 0.25**	99.10 ± 0.13
(c) Antikaon identification on complete data only			
Delete	95.82 ± 0.69	96.84 ± 0.66	96.33 ± 0.11
Ensemble	**96.14 ± 0.32**	97.60 ± 0.24	96.87 ± 0.09
Mean	**96.14 ± 0.45**	94.77 ± 0.99	95.45 ± 0.33
Proposed	96.00 ± 0.11	**97.85 ± 0.12**	**96.91 ± 0.11**
Regression	93.85 ± 01.11	96.41 ± 0.18	95.11 ± 0.58
(d) Antipion identification on data including incomplete examples			
Standard	**99.99 ± 0.01**	78.03 ± 0.01	87.66 ± 0.87
Ensemble	97.01 ± 0.87	**99.56 ± 0.11**	98.27 ± 0.42
Mean	97.23 ± 0.03	99.47 ± 0.03	98.34 ± 0.01
Proposed	97.38 ± 0.04	99.51 ± 0.02	**98.44 ± 0.02**
Regression	97.22 ± 0.15	99.52 ± 0.12	98.36 ± 0.03
(e) Antiproton identification on data including incomplete examples			
Standard	**99.24 ± 0.01**	53.02 ± 0.03	69.12 ± 1.93
Ensemble	96.90 ± 0.24	**92.41 ± 0.21**	94.60 ± 0.10
Mean	97.24 ± 0.39	92.39 ± 0.15	94.75 ± 0.20
Proposed	97.51 ± 0.55	92.40 ± 0.74	**94.89 ± 0.14**
Regression	96.89 ± 0.50	92.36 ± 0.22	94.57 ± 0.13
(f) Antikaon identification on data including incomplete examples			
Standard	**92.22 ± 0.01**	55.68 ± 0.04	69.44 ± 1.60
Ensemble	89.16 ± 01.51	81.06 ± 1.74	84.91 ± 0.48
Mean	89.75 ± 0.80	80.14 ± 1.18	84.67 ± 0.38
Proposed	90.86 ± 0.70	**81.64 ± 0.63**	**86.00 ± 0.13**
Regression	91.63 ± 0.58	79.29 ± 0.59	85.01 ± 0.13

data (including incomplete examples) in the most challenging kaon detection task. This plot provides detailed comparison between methods without a need for threshold selection. Additionally, in Fig. 5, we analyze the differences in performance for different range of particle momentum p_T. We can observe that our approach yields significantly higher area under the curve performance, and has a lower degradation in performance given particles with high momentum values.

In Figs. 6 and 7, we perform similar analysis for using only complete test data examples. Detailed evaluations for all other particles are included in the supplementary material.

4.5 Computational time comparison

In Tables 6 and 7, we present the computational overhead of calculating predictions with our method, when compared to the other baselines. In particular for training time, we report average time needed (in milliseconds) to update model's

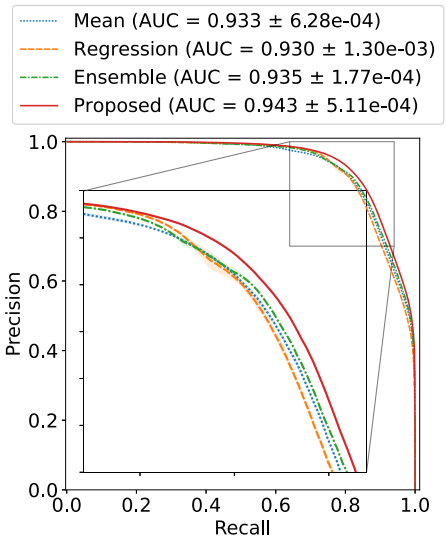

Fig. 4 Precision recall curve for different ML based approaches with missing data

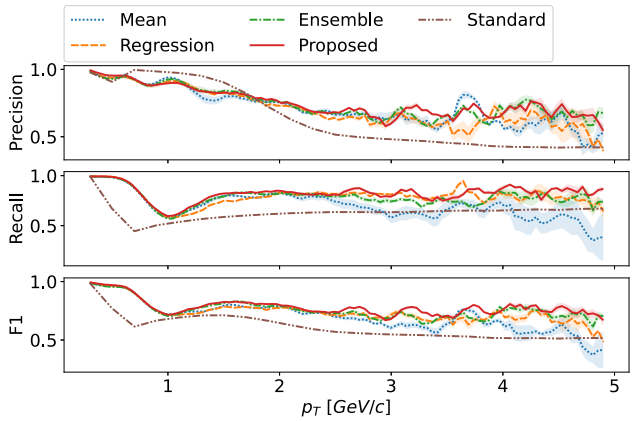

Fig. 5 Performance of different PID methods in kaon selection task with missing data as a function of particle momentum

weights from loading data, through forward and backward propagation up to the final update. For inference time, we report the average forward pass trough the network (in microseconds). Since our model requires calculation of the attention between features coming from different detectors, both it's training and inference times are around 3 times longer than for a standard methods. However, all methods can be easily parallelized (up to the GPU memory size), what can be seen in nearly constant times across different batch sizes. All of the calculations are performed on a small local machine with Nvidia GTX 1660Ti 6GB GPU, Intel core i7-9750 H CPU, and 16 GB 2400 MHz (MT/s) of RAM.

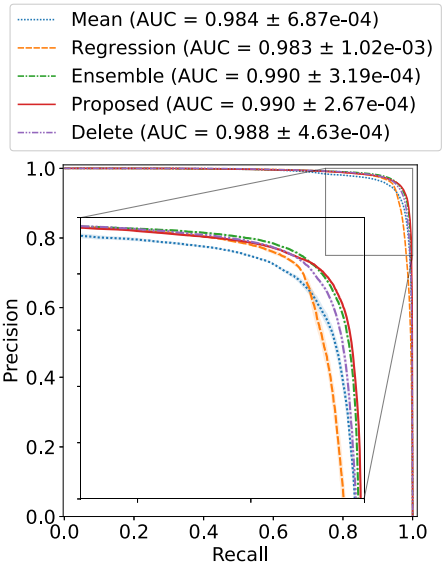

Fig. 6 Precision recall curve for different ML based approaches without missing data

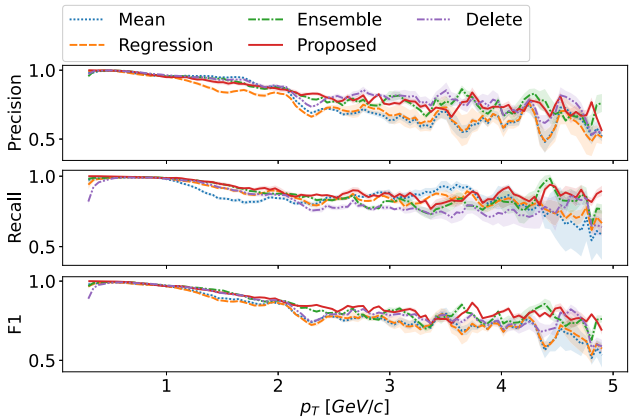

Fig. 7 Performance of different PID methods in kaon selection task without missing data as a function of particle momentum

Table 6 Average training time of our method compared to the baseline [ms]

Batch size	64	128	256	512
Mean	3.10	3.54	3.69	3.64
Regression	2.98	3.18	3.37	3.40
Ensemble	3.67	3.62	3.74	3.99
Proposed	11.09	10.21	10.82	12.65
Delete	3.36	3.31	3.39	3.68

Table 7 Average training time of our method compared to the baseline [μs]

Batch size	64	128	256	512
Mean	413.1	364.0	355.7	365.3
Regression	370.5	349.2	358.9	377.2
Ensemble	415.1	411.6	436.8	420.7
Proposed	1507.7	1561.9	1592.8	1530.8
Delete	479.0	462.3	432.8	462.5

5 Conclusion

This work considers the real-case scenario of classification from incomplete data in the particle identification task, where due to the nature of physical processes, not all of the information is always recorded by all of the detectors. To solve this problem, we propose a novel method based on the attention mechanism. We verify that our approach is able to learn from both complete and incomplete data improving the performance of models trained solely on the complete examples. Moreover, our method provides no worse performance than other techniques while avoiding their drawbacks such as insertion of artificial data (imputation methods) and potentially too complex architecture (neural network ensemble).

Acknowledgements We would like to thank the ALICE Collaboration for guidance and support during our research as well as for the access to all software and data. This work was supported by the Polish National Science Centre under agreements no. no. 2021/43/D/ST2/02214 and UMO-2022/45/B/ST2/02029, by the Polish Ministry for Education and Science under agreements no. 2022/WK/01 and 5236/CERN/2022/0, as well as by the IDUB-POB-FWEiTE-2 project granted by Warsaw University of Technology under the program Excellence Initiative: Research University (ID-UB).

Data Availability Statement This manuscript has no associated data. [Author's' comment: It is not possible to share the data used for this study outside of the ALICE Collaboration.]

Code Availability Statement This manuscript has associated code/software in a data repository. [Author's comment: Code repository link: https://github.com/KamilDeja/PID_with_missing_data.]

Funded by SCOAP[3].

References

1. K. Aamodt et al., The ALICE experiment at the CERN LHC. J. Instrum. **3**, 08002 (2008). https://doi.org/10.1088/1748-0221/3/08/S08002
2. L. Evans, P. Bryant, LHC machine. J. Instrum. **3**, 08001 (2008). https://doi.org/10.1088/1748-0221/3/08/S08001
3. S. Acharya et al., The ALICE experiment—a journey through QCD. (2022). arXiv:2211.04384 [nucl-ex]
4. E. Botta, Particle identification performance at ALICE, in *Proceeding of the Fifth Annual Conference on Large Hadron Collider Physics* (2017). arXiv:1709.00288 [nucl-ex]
5. G. Dellacasa et al., ALICE time projection chamber: technical design report. Technical design report. ALICE. CERN, Geneva (2000). https://cds.cern.ch/record/451098. Accessed 12 Dec 2023
6. G. Dellacasa et al., ALICE time-of-flight system (TOF): technical design report. Technical design report. ALICE. CERN, Geneva (2000). https://cds.cern.ch/record/430132. Accessed 12 Dec 2023
7. P. Cortese, ALICE transition-radiation detector: technical design report. Technical design report. ALICE. CERN, Geneva (2001). https://cds.cern.ch/record/519145. Accessed 12 Dec 2023
8. J. Adam et al., Particle identification in ALICE: a Bayesian approach. Eur. Phys. J. Plus **131**(5), 168 (2016). https://doi.org/10.1140/epjp/i2016-16168-5. arXiv:1602.01392 [physics.data-an]
9. R. Aaij et al., LHCb detector performance. Int. J. Mod. Phys. A **30**(07), 1530022 (2015). https://doi.org/10.1142/S0217751X15300227
10. J. Collado et al., Learning to identify electrons. Phys. Rev. D **103**(11), 116028 (2021). https://doi.org/10.1103/PhysRevD.103.116028
11. A. Tumasyan et al., Identification of hadronic tau lepton decays using a deep neural network. J. Instrum. **17**, 07023 (2022). https://doi.org/10.1088/1748-0221/17/07/P07023
12. Z. Wang, J. Poon, S. Sun, S. Poon, Attention-based multi-instance neural network for medical diagnosis from incomplete and low quality data, in *2019 International Joint Conference on Neural Networks (IJCNN)* (2019), pp. 1–8. https://doi.org/10.1109/IJCNN.2019.8851846
13. P.J. García-Laencina, J.-L. Sancho-Gómez, A.R. Figueiras-Vidal, Pattern classification with missing data: a review. Neural Comput. Appl. **19**(2), 263–282 (2010). https://doi.org/10.1007/s00521-009-0295-6
14. Z. Ghahramani, M.I. Jordan: Learning from incomplete data. Technical report AIM-1509, Massachusetts Institute of Technology (1995)
15. R.J.A. Little, D.B. Rubin, *Statistical Analysis with Missing Data*, 2nd edn. (Wiley, New York, 2014), pp.11–19
16. A. Jadhav, D. Pramod, K. Ramanathan, Comparison of performance of data imputation methods for numeric dataset. Appl. Artif. Intell. **33**(10), 913–933 (2019). https://doi.org/10.1080/08839514.2019.1637138
17. W. Young, G. Weckman, W. Holland, A survey of methodologies for the treatment of missing values within datasets: limitations and benefits. Theor. Issues Ergon. Sci. **12**(1), 15–43 (2011). https://doi.org/10.1080/14639220903470205
18. K. Jiang, H. Chen, S. Yuan, Classification for incomplete data using classifier ensembles, in *2005 International Conference on Neural Networks and Brain*, vol. 1 (2005), pp. 559–563. https://doi.org/10.1109/ICNNB.2005.1614675
19. P. Juszczak, R.P.W. Duin, Combining one-class classifiers to classify missing data, in *International Workshop on Multiple Classifier Systems* (2004), pp. 92–101. https://doi.org/10.1007/978-3-540-25966-4_9
20. S. Krause, R. Polikar, An ensemble of classifiers approach for the missing feature problem, in *Proceedings of the International*

Joint Conference on Neural Networks, vol. 1 (2003), pp. 553–558. https://doi.org/10.1109/IJCNN.2003.1223406
21. P.K. Sharpe, R. Solly, Dealing with missing values in neural network-based diagnostic systems. Neural Comput. Appl. **3**(2), 73–77 (1995). https://doi.org/10.1007/BF01421959
22. M.L. Morvan, J. Josse, E. Scornet, G. Varoquaux, What's a good imputation to predict with missing values? arXiv preprint (2021). arXiv:2106.00311
23. Z.C. Lipton, D.C. Kale, R. Wetzel et al., Modeling missing data in clinical time series with RNNs. Mach. Learn. Healthc. **56**, 253–270 (2016)
24. J.M. Jerez, I. Molina, P.J. García-Laencina, E. Alba, N. Ribelles, M. Martín, L. Franco, Missing data imputation using statistical and machine learning methods in a real breast cancer problem. Artif. Intell. Med. **50**(2), 105–115 (2010)
25. W. Shi, Y. Zhu, J. Zhang, X. Tao, G. Sheng, Y. Lian, G. Wang, Y. Chen, Improving power grid monitoring data quality: an efficient machine learning framework for missing data prediction, in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems* (IEEE, 2015), pp. 417–422
26. Z. Cui, R. Ke, Z. Pu, Y. Wang, Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values. Transp. Res. Part C Emerg. Technol. **118**, 102674 (2020)
27. D. Grangier, I. Melvin, Feature set embedding for incomplete data. Adv. Neural Inf. Process. Syst. **23**, 793–801 (2010)
28. A. Vaswani et al., Attention is all you need. Adv. Neural Inf. Process. Syst. **30**, 5998–6008 (2017)
29. T. Sjöstrand et al., An introduction to PYTHIA 8.2. Comput. Phys. Commun. **191**, 159–177 (2015). https://doi.org/10.1016/j.cpc.2015.01.024. arXiv:1410.3012 [hep-ph]
30. R. Brun et al., GEANT Detector Description and Simulation Tool. Technical report, CERN (1994). https://doi.org/10.17181/CERN.MUHF.DMJ1