Faculty of Physics
WARSAW UNIVERSITY OF TECHNOLOGY

Particle identification with machine learning in ALICE Run 3

Maja Kabus,
Monika Jakubowska, Kamil Deja,
Łukasz Graczykowski,
Miłosz Kasak

on behalf of the ALICE Collaboration

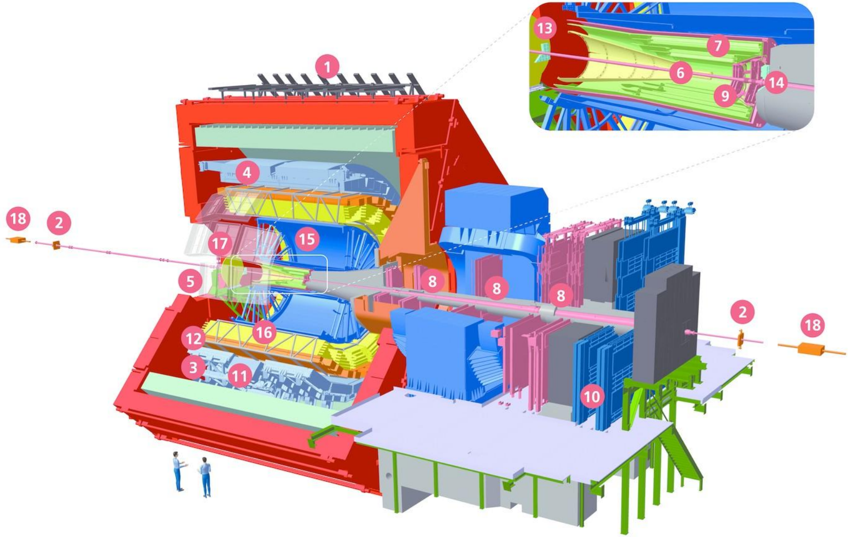
CHEP, 05.2023

ALICE

CERN

The ALICE experiment

ALICE – one of the experiments at the **Large Hadron Collider (LHC)** at CERN



LHC Run 1+2 configuration

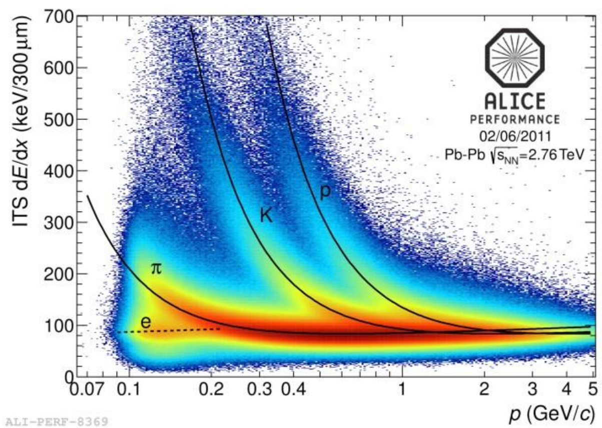
Heavy-ion collisions → production of **quark-gluon plasma (QGP)**

- beginnings of the Universe
- neutron stars

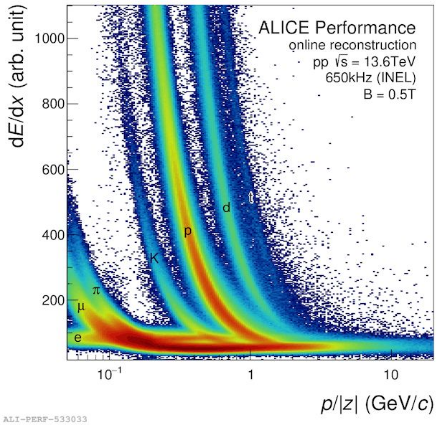
Particle identification (PID)

Aim: provide high purity samples of particles of a given type

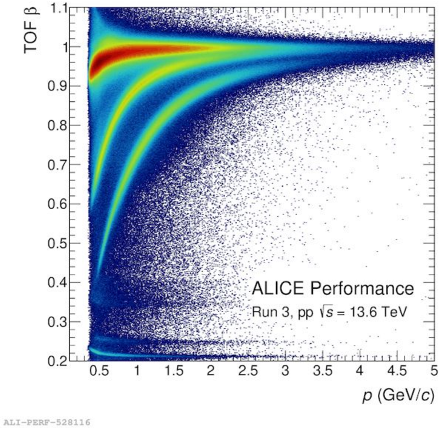
- **an essential step** for many physics analyses, especially **quark-gluon plasma** measurements
- **a distinguishing feature** of ALICE among the LHC experiments:
  - identification of particles of momenta from **100 MeV/*c* up to 20 GeV/*c***
  - **very good separation** of pion, kaons, protons, electrons
  - **all known techniques** employed: dE/dx energy loss, time-of-flight, Cherenkov radiation for hadrons and transition radiation for electrons



ITS



TPC

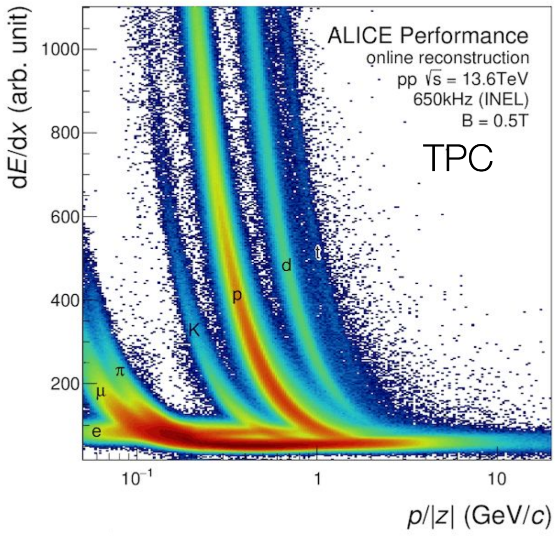


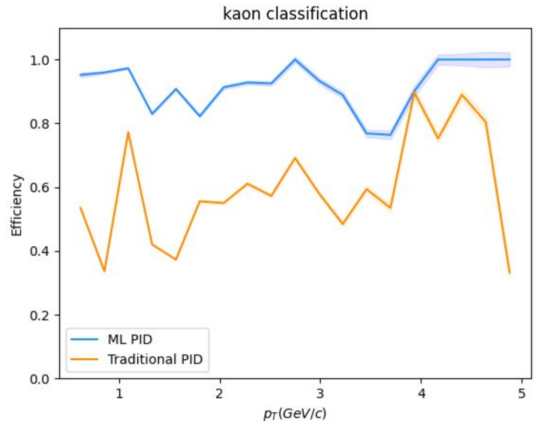
TOF

Present state-of-art

1. **Traditional method:**
   - hand-crafted selections of selected quantities, e.g., nσ
   - problems:
     - overlapping signals
     - time-consuming optimization

2. **Bayesian method ([arxiv:1602.01392](https://arxiv.org/abs/1602.01392)):**
   - updating probability of an hypothesis with each new evidence
   - priors = best guess of true particle yields per events
   - posteriors ~ purity
   - increased purity, results consistent with the traditional method

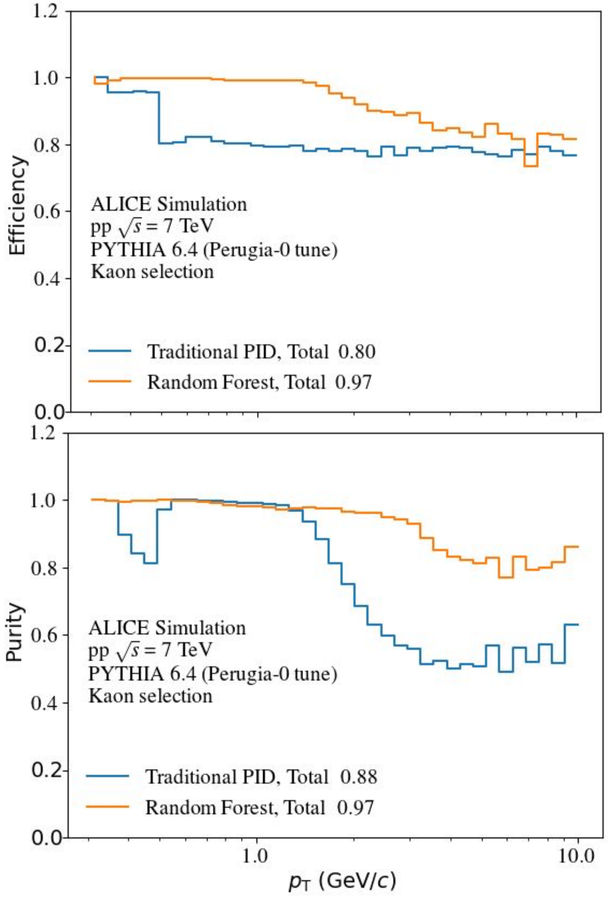




Both methods available in O^{2}Physics – ALICE Run 3 software:
https://aliceo2group.github.io/analysis-framework/

Can we go any better?
nσ method: high purity above 0.9 at the cost of low efficiency → can we balance?

Machine learning for PID



- **classification** problem – a ML "standard"
- can use more track parameters as input
- can learn **more complex relationships**
- many software libraries available

Note also **the limitations**:

- good quality of the training data
- hard to obtain systematic uncertainties
- hard to follow classifier's "reasoning"

Machine learning can **greatly improve** purity and efficiency

- random forest: T. Trzciński, Ł. Graczykowski, M. Glinka, ALICE Collaboration. Using Random Forest classifier for particle identification in the ALICE experiment. Conference on Information Technology, Systems Research and Computational Physics, pp. 3-17. 2018
- domain adaptation: M. Kabus, M. Jakubowska, Ł. Graczykowski, K. Deja, ALICE Collaboration. Using machine learning for particle identification in ALICE. JINST, v. 17, p. C07016. 2022
- details in backup

Dealing with incomplete data

At present: simple neural network, **19 features:** momenta, spatial coordinates, charge sign, DCA XY, DCA Z, alpha angle, track type, TPC shared clusters, detector signals

Data might be missing from one or more detectors due to, e.g., too small p_T
Challenge: Classify without making any assumptions about the missing values

Feature Set Embedding (article):

- instead of vectors, use (feature, value) pairs; no value → no pair
- map pairs into an embedding space of fixed dimension: similar features close to each other
- predict output class from embedded vectors
- **2 functions (networks) to learn:** (feature, value) pairs→embeddings, embeddings→class

Bonus: simultaneous learning of variants with and without given feature

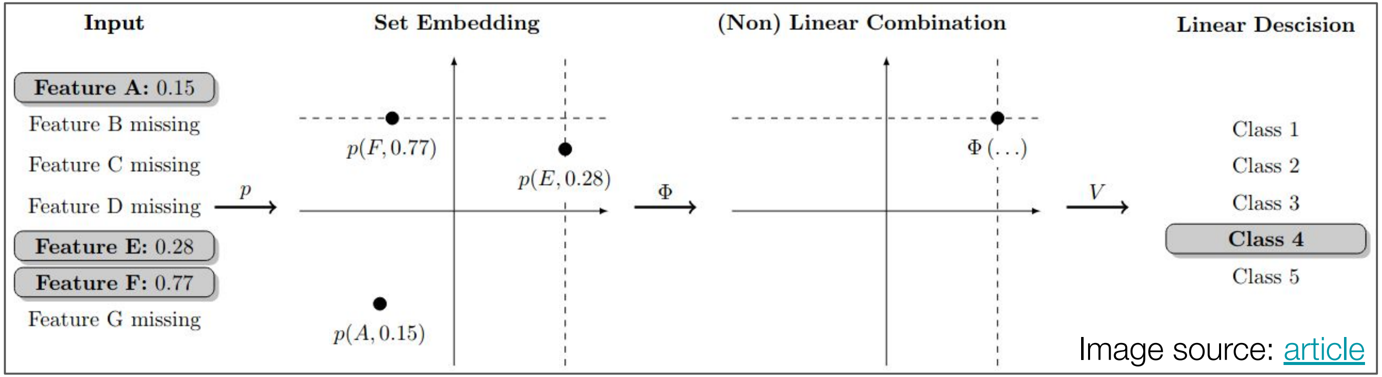
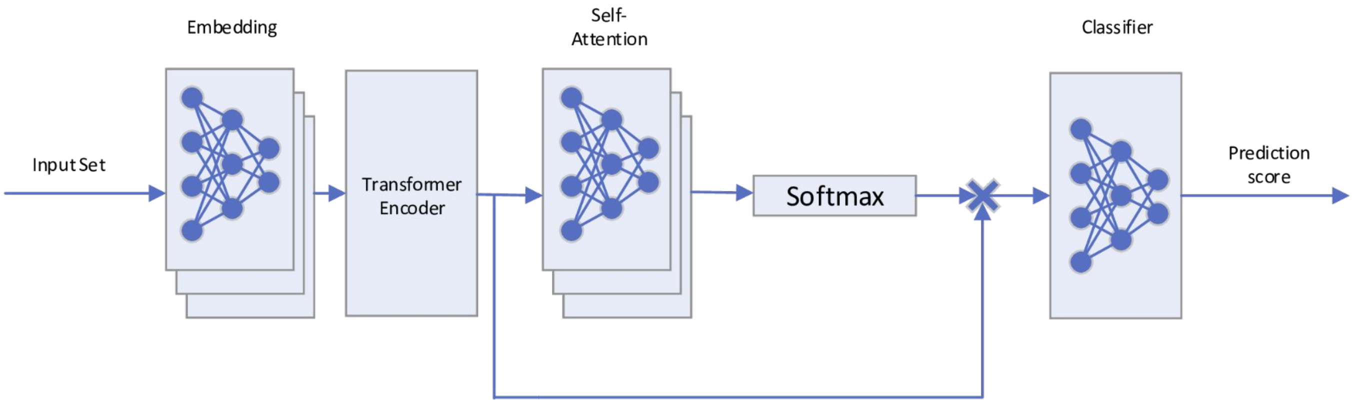


Image source: article

One step further: the attention mechanism

Inspired by AMI-Net proposed for medical diagnosis from incomplete data



1. Feature Set Embedding to encode the inputs
   a. one-hot encoding of feature indices for easier processing
2. Transformer Encoder to detect patterns in the input
3. Self-attention to pool the encoder output set into a single vector
4. **Classifier:** a simple neural network for **a specific particle specie**
   a. "certainty" in range (0, 1) that a given particle belongs to the given specie

details in backup

Test setup

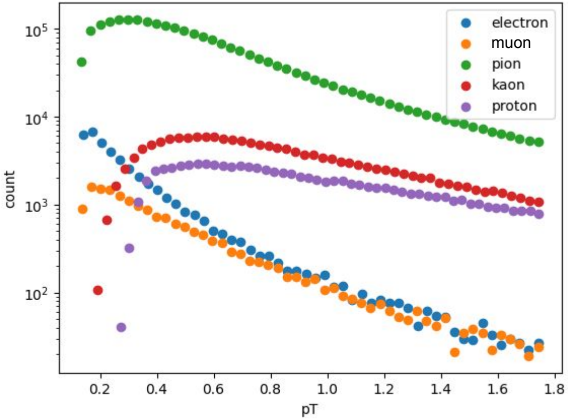
5 methods for incomplete data:

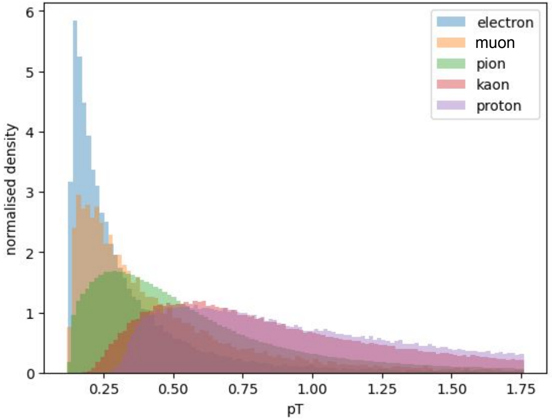
- imputation
  - mean
  - linear regression
- case deletion
- neural networks ensemble
- attention + FSE

architecture details in backup

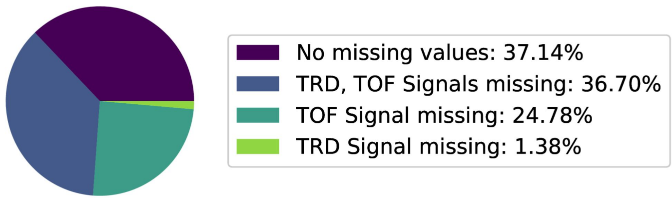
Hyperparameter sweep to choose best model for each method

Dataset: Run 2 general-purpose MC pp at $\sqrt{s}$ = 13 TeV simulated with Pythia8 and Geant4





Missing data distribution



Results – particles

F_1 = (purity x efficiency) / (purity + efficiency), **best model**, **2nd best model**

FSE + attention with **very good scores** of **F_1, purity and efficiency**

No flaws of other methods:

- imputation:
  artificial bias in data
- case deletion:
  no ability to analyze samples with missing detector signals
- NN ensemble:
  potentially large complexity

	π			p			K		
model	purity	efficiency	F_1	purity	efficiency	F_1	purity	efficiency	F_1
mean	0.9718	0.9934	0.9825	0.9559	0.8927	0.9232	0.8858	0.8081	0.8452
regression	0.9723	0.9931	0.9826	0.9520	0.8973	0.9238	0.8795	0.8168	0.8470
case deletion	–	–	–	–	–	–	–	–	–
NN ensemble	0.9745	0.9914	0.9829	0.9607	0.8895	0.9237	0.8751	0.8207	0.8470
attention + FSE	0.9734	0.9937	0.9835	0.9648	0.9009	0.9318	0.8841	0.8337	0.8581

	π, only complete data			p, only complete data			K, only complete data		
model	purity	efficiency	F_1	purity	efficiency	F_1	purity	efficiency	F_1
mean	0.9862	0.9945	0.9904	0.9817	0.9737	0.9777	0.9210	0.9334	0.9272
regression	0.9885	0.9920	0.9903	0.9721	0.9841	0.9781	0.9043	0.9450	0.9242
case deletion	0.9884	0.9946	0.9915	0.9715	0.9840	0.9777	0.9449	0.9365	0.9407
NN ensemble	0.9895	0.9929	0.9912	0.9757	0.9818	0.9787	0.9272	0.9530	0.9399
attention + FSE	0.9884	0.9941	0.9913	0.9799	0.9825	0.9812	0.9249	0.9595	0.9419

Results – antiparticles

F_1 = (purity x efficiency) / (purity + efficiency), **best model**, **2nd best model**

FSE + attention with **very good scores** of **F_1, purity and efficiency**

No flaws of other methods:

- imputation:
  artificial bias in data
- case deletion:
  no ability to analyze samples with missing detector signals
- NN ensemble:
  potentially large complexity

	π^-			$\bar{p}$			$\bar{K}$		
model	purity	efficiency	F_1	purity	efficiency	F_1	purity	efficiency	F_1
mean	0.9710	0.9928	0.9818	0.9352	0.8815	0.9076	0.8531	0.8013	0.8264
regression	0.9716	0.9924	0.9819	0.9415	0.8773	0.9082	0.8715	0.7914	0.8295
case deletion	–	–	–	–	–	–	–	–	–
NN ensemble	0.9725	0.9928	0.9826	0.9528	0.8717	0.9105	0.8578	0.8174	0.8371
attention + FSE	0.9727	0.9939	0.9831	0.9579	0.8805	0.9176	0.8870	0.8059	0.8445

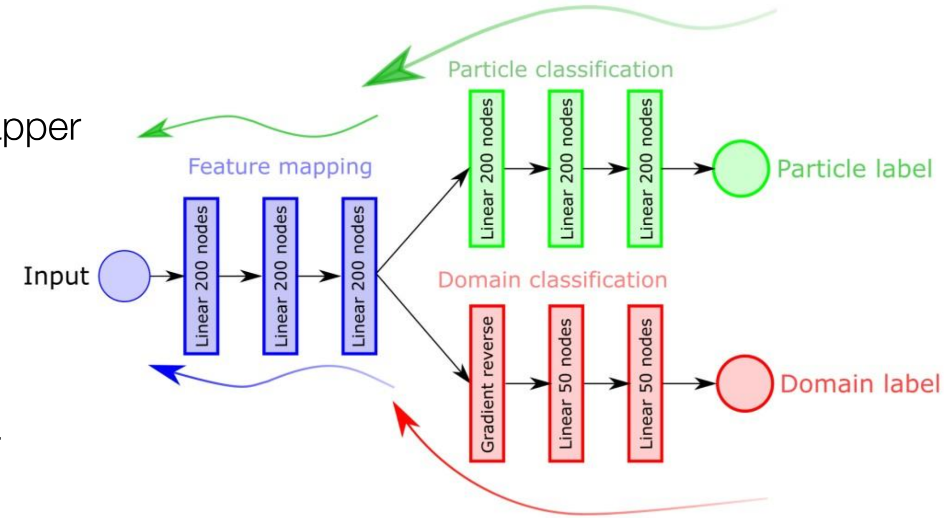
	π^-, only complete data			$\bar{p}$, only complete data			$\bar{K}$, only complete data		
model	purity	efficiency	F_1	purity	efficiency	F_1	purity	efficiency	F_1
mean	0.9869	0.9940	0.9905	0.9693	0.9695	0.9694	0.8990	0.9329	0.9156
regression	0.9889	0.9913	0.9901	0.9565	0.9747	0.9655	0.8962	0.9437	0.9193
case deletion	0.9886	0.9942	0.9914	0.9663	0.9687	0.9675	0.9324	0.9226	0.9275
NN ensemble	0.9887	0.9944	0.9915	0.9668	0.9745	0.9707	0.9216	0.9488	0.9350
attention + FSE	0.9885	0.9950	0.9918	0.9731	0.9758	0.9745	0.9326	0.9434	0.9380

Domain Adversarial Neural Networks (DANNs)

feature mapping: input → domain invariant features
particle classifier: recognize particles based on domain invariant latent space
domain classifier: recognize MC vs real samples

Training more complicated:

1. Train the domain classifier independently.
2. Freeze the domain classifier.
3. Train jointly particle classifier and feature mapper **adversarially** to the domain classifier.
4. Weights of the feature mapper:
   gradient from particle classifier
   + reversed gradient from domain classifier



Application time similar to a standard classifier

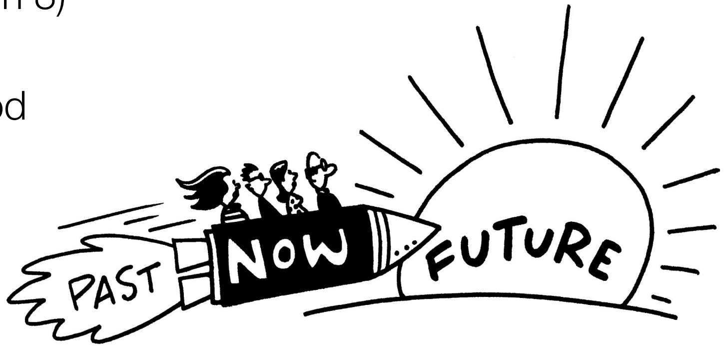
Summary and outlook

Summary:

- **machine learning** is a promising way to identify particles with **higher purity and efficiency**
- **Feature Set Embedding with Multi-Head Attention** improve F_1 score for PID on **incomplete data**

Plans:

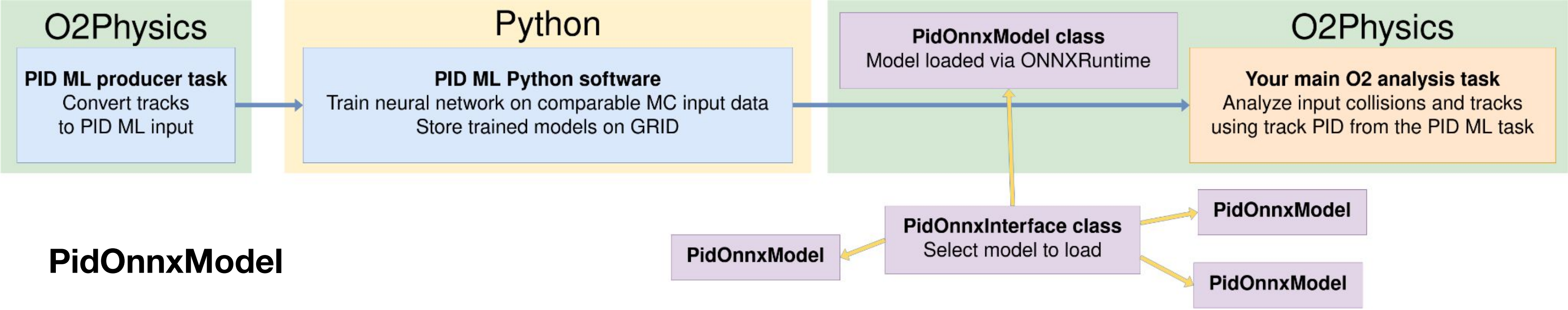
- test in an analysis task
- test on MC data from the next LHC data-taking period (Run 3)
- add domain adaptation and test on the new real data
- regular production of models for the new data-taking period



Thank you for your attention!

Backup

Integration with O^{2}Physics: user interface



PidOnnxModel

- 1 instance = 1 model = 1 particle specie recognized (yes / no)
- **convenient interface** clearly separated from the rest of analysis
- using all capabilities of **Python ML libraries** for training
- ONNX file format and **ONNXRuntime** software used for inference in O^2 C++ environment

PidOnnxInterface

- **automatically select most suitable model** for user needs or manual mode
- as **little additional knowledge** from the analyser as possible

Random Forest (RF) on Run 2 data

Preliminary work in 2019 for LHC Run 2

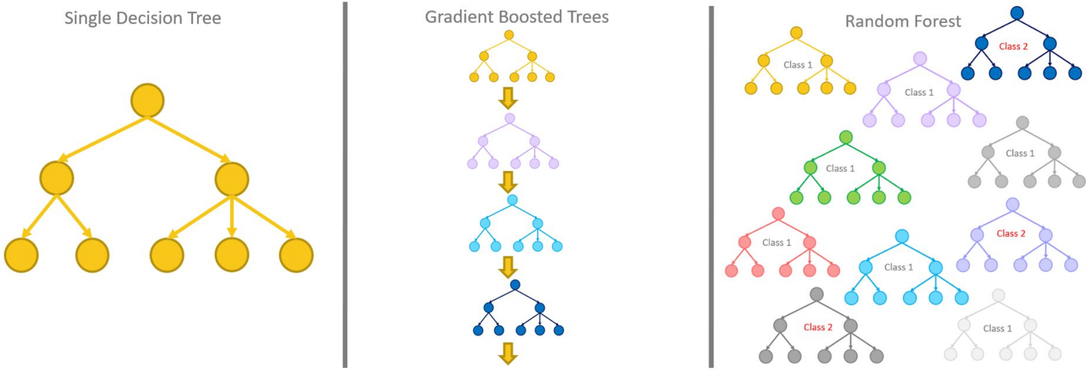
Tomasz Trzciński, Łukasz Graczykowski, Michał Glinka, ALICE Collaboration, et al. Using Random Forest classifier for particle identification in the ALICE experiment. In Conference on Information Technology, Systems Research and Computational Physics, pages 3–17. Springer, 2018

Why Random Forest?

- a set of decision trees, each trained on a random subset of the training data
- easy to parallelize, e.g., on GRID
- resistant to overfitting

Our approach

- tree generation: Gini index
- selection: majority of votes by trees
- adaptive boosting



Run 2 results

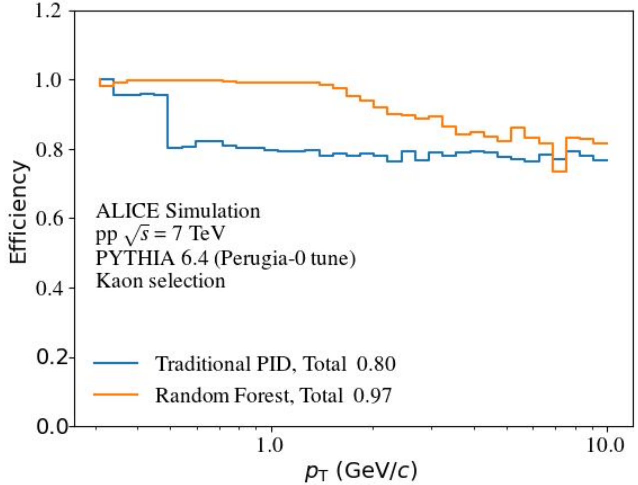
- pp at 7 TeV, Pythia 6 Perugia-0
- kaons vs other particles

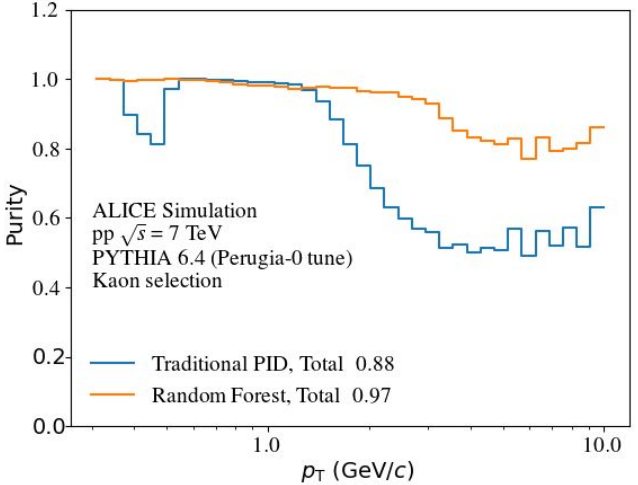
Traditional PID:

$$n^2_{\sigma,\mathrm{TPC}} \qquad p_\mathrm{T} \leq 0.5\mathrm{GeV/c}$$

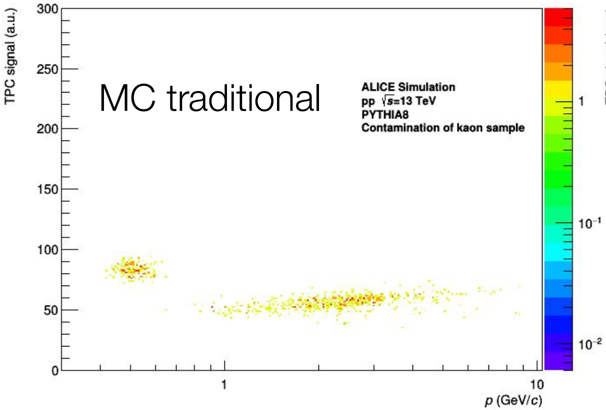
$$\sqrt{n^2_{\sigma,\mathrm{TPC}} + n^2_{\sigma,\mathrm{TOF}}} \qquad p_\mathrm{T} > 0.5\mathrm{GeV/c}$$

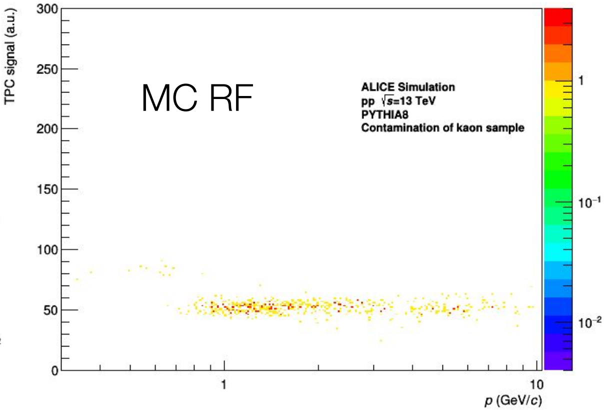
much higher efficiency and **purity** with Random Forest





Contamination of kaon samples

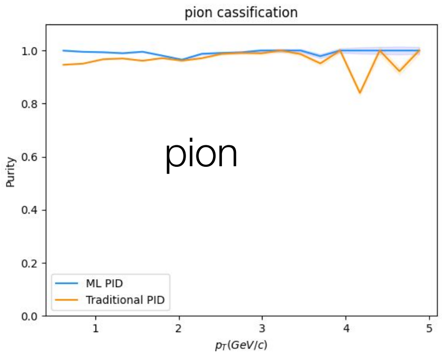


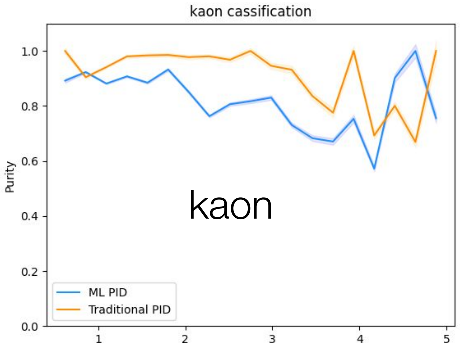


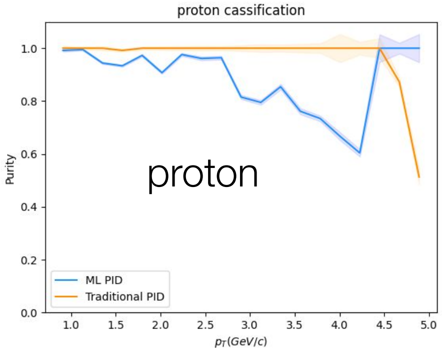
Baseline – plain vanilla neural networks

- one neural network model per particle and per set of detectors
- results for using all detectors; **ML PID**, **traditional approach**

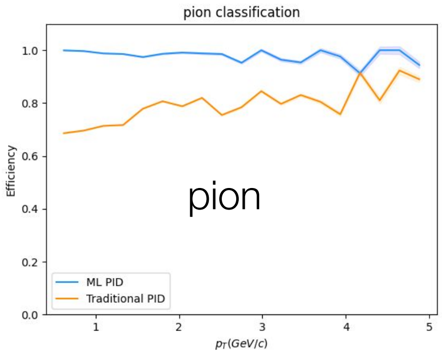
purity = precision / specificity

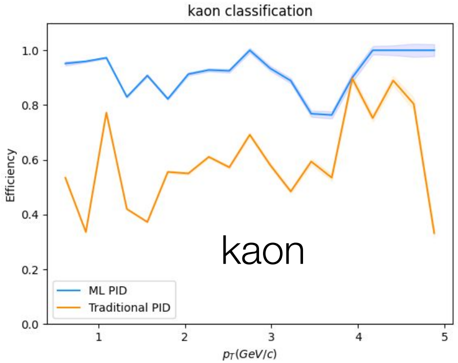


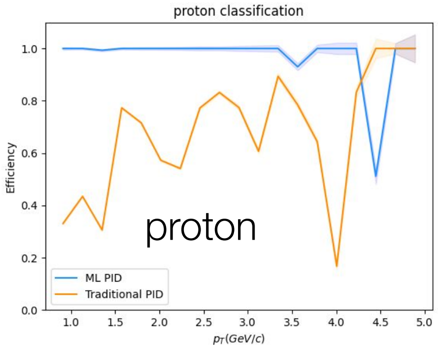




efficiency = recall / sensitivity





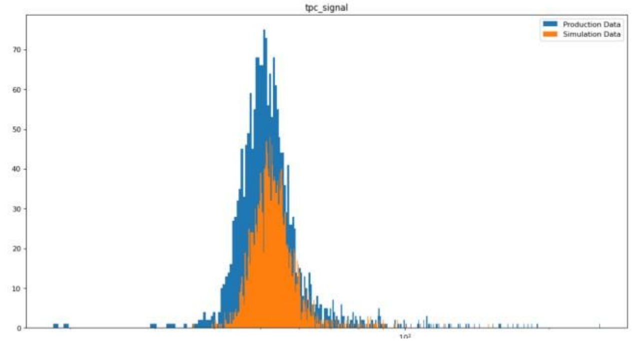


Domain adaptation

Training set: labeled data → MC samples

Apply set: unlabeled real data with **different distributions of attributes**

→ worse performance on real data



How can we transfer the knowledge from training to inference?

Standard PID example: **"tune on data"**

- get parametrization from data → real data
- generate a random detector signal → MC data
- equivalent distributions of real and MC samples – the differences are statistical fluctuations

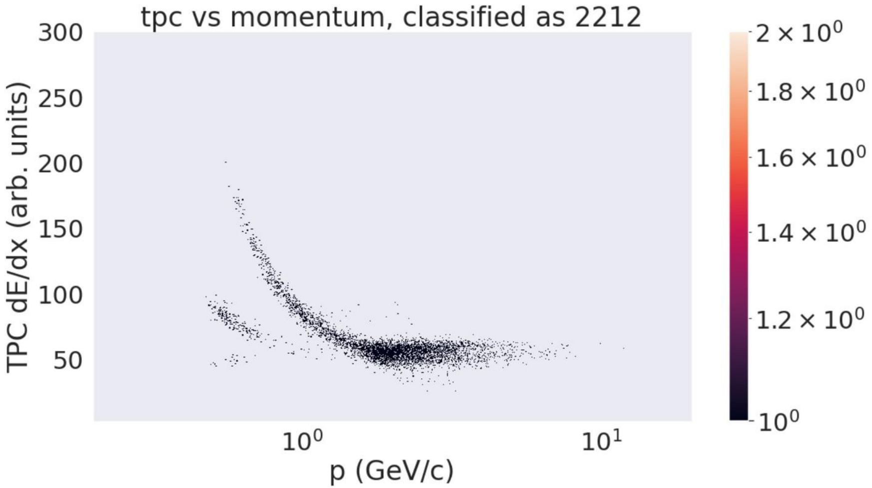
Machine learning:

- actually **learn** the difference between data domains
- translate both data to a single common hyperspace

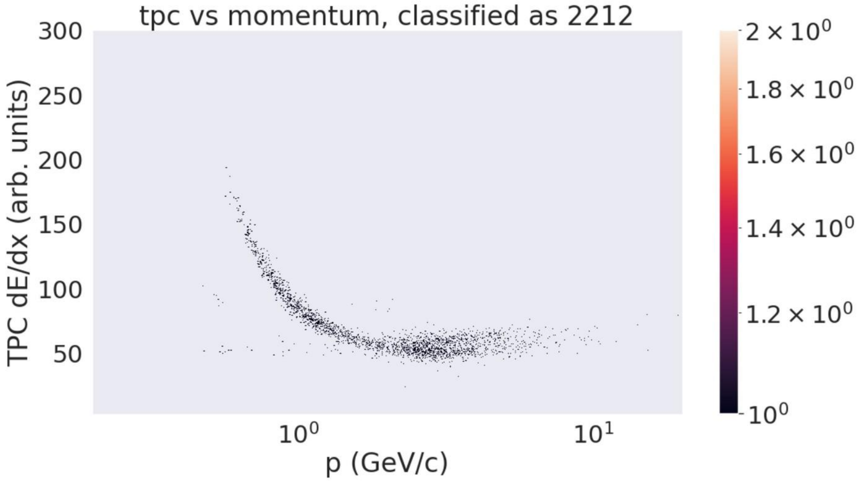
First results of domain adaptation

- pp data at 13 TeV, LHC Run 2
- training: PYTHIA 8 with Monash tune
- classification improved – **reduction of contamination**
- more research ongoing

No domain adaptation



With domain adaptation



Simple network implementation

- linear layers with Leaky ReLU, sigmoid at the end
- simple: dropout after each linear layer

Parameters:

- optimizer: Adam
- output layer: 1 node (yes / no for a given particle)
- loss function: binary cross entropy
- scheduler: exponential with rate 0.98
- learning rate: 0.0005
- batch size: 64
- epochs: 30

Example: FSE with one-hot encoding

From the article in preparation

Table 1: Preprocessing of data samples into feature set values – example.

(a) 3 data samples with 5 attributes with different amount of missing values.

id	momentum	TOF	TPC	TRD	ITS
1	0.1		3		5
2	7	70	24	13	88
3		78			

(b) First particle

key					value
1	0	0	0	0	0.1
0	0	1	0	0	3
0	0	0	0	1	5

(c) Second particle.

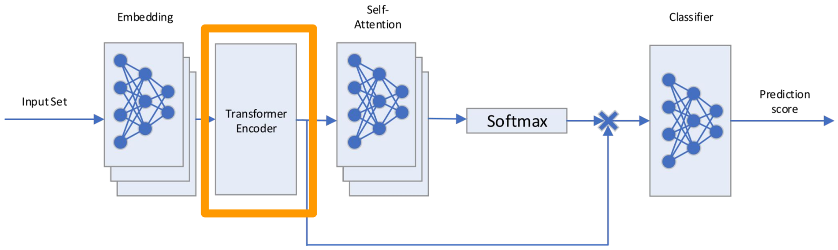
key					value
1	0	0	0	0	7
0	1	0	0	0	70
0	0	1	0	0	24
0	0	0	1	0	13
0	0	0	0	1	88

(d) Third particle.

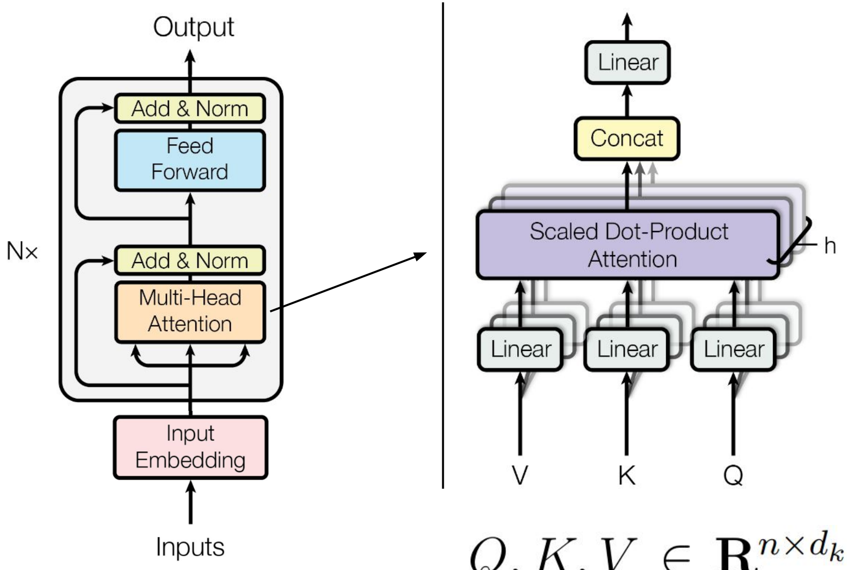
key					value
0	1	0	0	0	78

The attention continued

2. Transformer Encoder



N=2



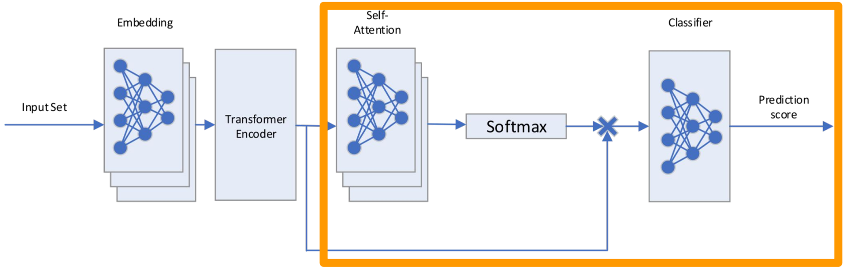
- adjusted original Transformer Encoder
- attention without convolutions and recurrence
- finding self-correlations in an instance set of vectors
- example: a specific detector signal could be used if and only if the momentum is in a specific range

$$Q, K, V \in \mathbf{R}^{n \times d_k}$$

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

modified diagram from the article

Pooling and final classification



Classifier: a simple neural network
expects a single vector as an input

Solution: self-attention to pool the variable-size vector set from Transformer Encoder

$$\{v_1, v_2, ..., v_n\},\ v_i \in \mathbf{R}^{d_{model}}$$

$$e_i = NN(v_i) \qquad \forall i \in [1, n]$$ self-attention values

$$\alpha'_j = softmax(e'_j) \qquad \forall j \in [1, d_{model}]$$ self-attention weights

$$o_j = \sum_{k=1}^{n} \alpha_{kj} v_{kj} \qquad \forall j \in [1, d_{model}]$$ pooled output vector

Classifier score: logistic function $f(x) = \frac{1}{1+e^{-x}}$, range (0, 1)
"certainty" that a given particle belongs to the given specie

Architecture of tested neural networks

Imputations, case deletion, and NN ensemble

- 3 hidden layers of sizes 64, 32, 16 with ReLU activation
- dropout 0.1 after each activation layer
- input size:
    - imputations and case deletion: 19 as all missing features are imputed
    - ensemble: 4 networks with input sizes 19, 17, 17, 15

Attention + FSE

- embedding layers: 20 – 128 – 32 neurons
- Transformer Encoder:
    - Multi-Head Attention: dimension 32, 2 heads
    - neural network layers: 32 – 128 – 32 neurons
    - 2 layers of Multi-Head Attention + neural network
- Self-Attention layers: 32 – 64 – 32 neurons
- classifier layers: 32 – 64 – 1 neurons
- dropout 0.1 at the output of embedding and each Transformer Encoder layer
- ReLU activation between neural network layers

Sample ROC curves

FSE+attention achieves **best results.**

Little variation between particle species.