Prototype of a cloud native solution of Machine Learning as Service for High Energy Physics

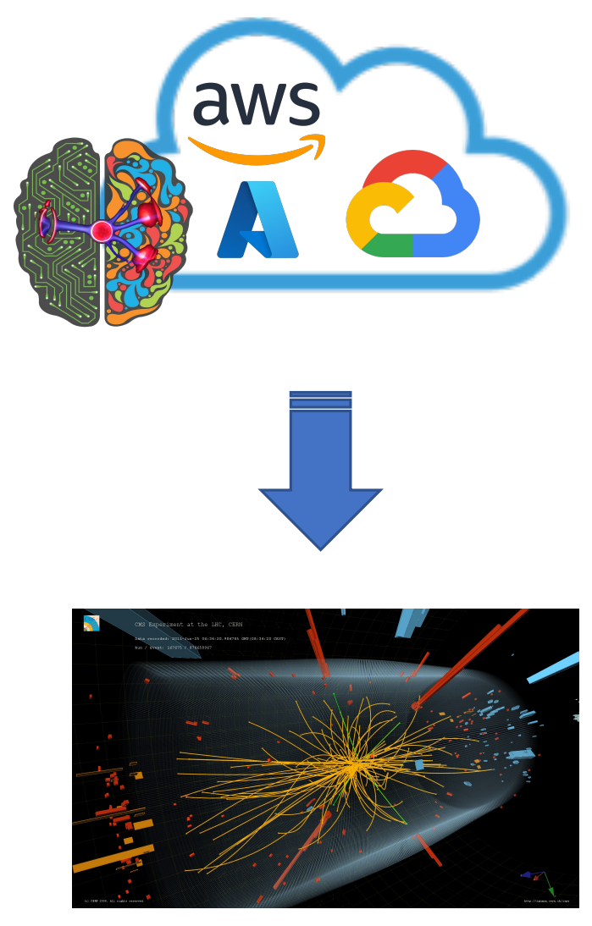
Luca Giommi[1], Daniele Spiga[2], Valentin Kuznetsov[3], Daniele Bonacorsi[1], on behalf of the CMS collaboration

[1]University of Bologna, INFN-Bologna, Italy
[2]INFN-Perugia, Italy
[3]Cornell University, Ithaca, NY, USA

Why Machine Learning and ML as a Service in High Energy Physics

Machine Learning (**ML**) techniques in the High Energy Physics (**HEP**) domain are ubiquitous, successfully used in many areas, and will play a significant role also in Run3 and High-Luminosity LHC upgrade.

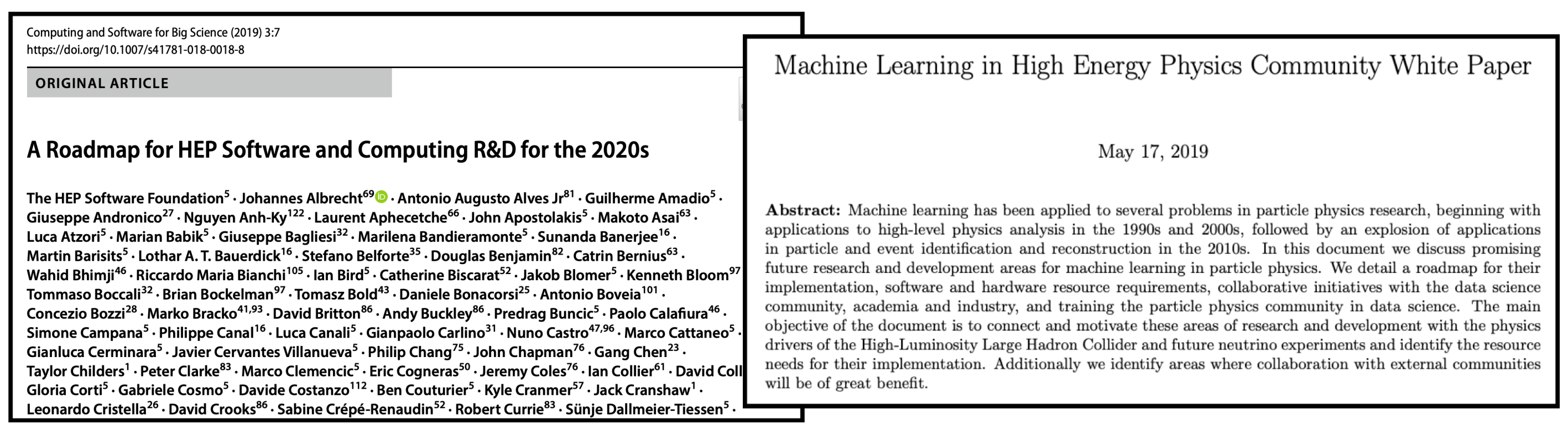
It is necessary to have a synergy between HEP and ML community and to ease the usage of ML techniques in HEP analyses. Therefore, it would be useful to provide physicists who are not experts in ML a service to easily exploit the ML potentiality.



Existing ML as a Service (**MLaaS**) solutions are many: they offer many services and cover different use cases but they are not directly usable in HEP.

Existing HEP R&D solutions don't cover the whole ML pipeline or they are not "aaS" solutions or they are difficult to generalize to other use cases.

We proposed a MLaaS for HEP (**MLaaS4HEP**) solution as a product of R&D activities within the CMS experiment [1].



Computing and Software for Big Science (2019) 3:7
https://doi.org/10.1007/s41781-018-0018-8

ORIGINAL ARTICLE

A Roadmap for HEP Software and Computing R&D for the 2020s

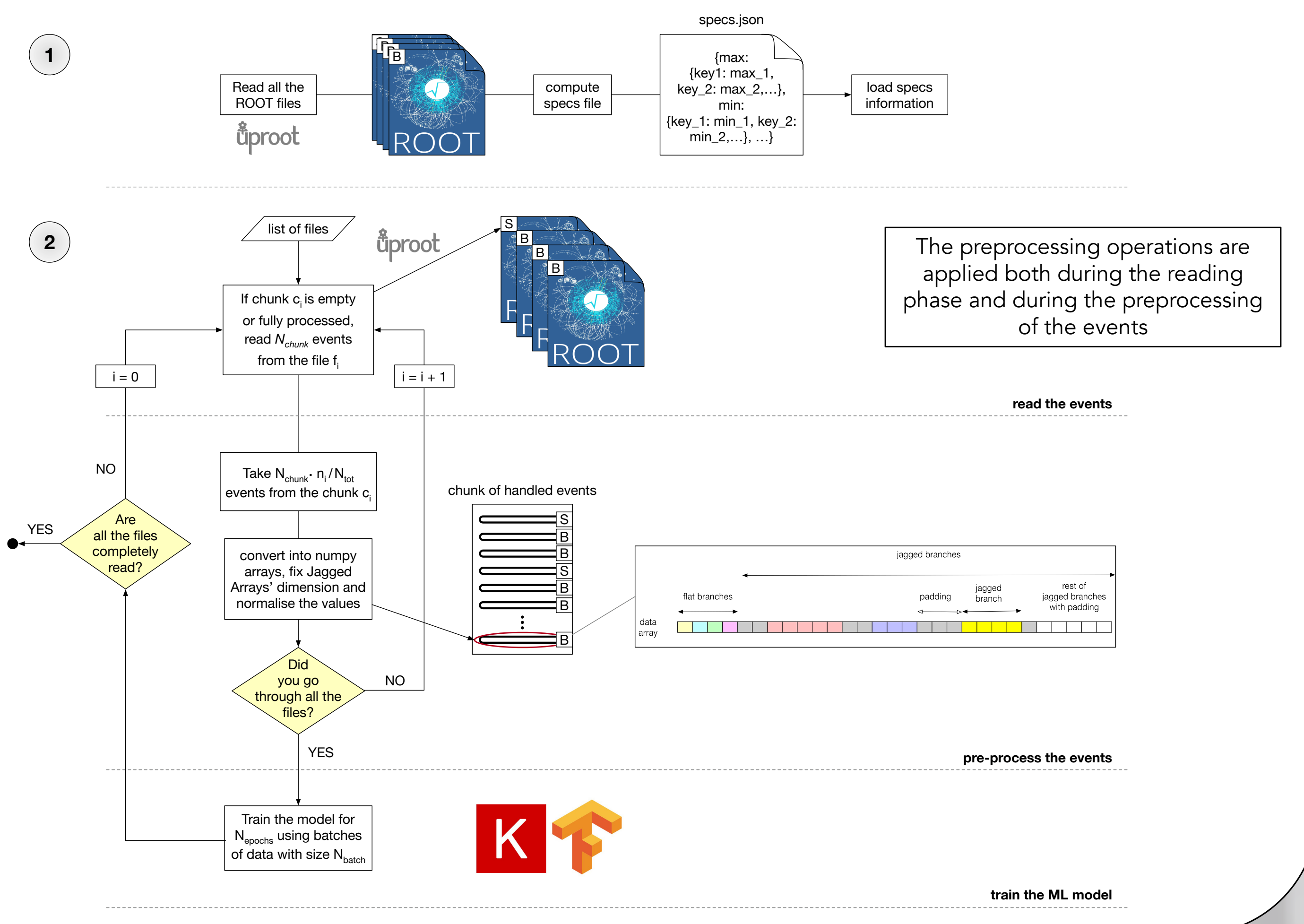
The HEP Software Foundation[5] · Johannes Albrecht[69] · Antonio Augusto Alves Jr[81] · Guilherme Amadio[5] · Giuseppe Andronico[27] · Nguyen Anh-Ky[122] · Laurent Aphecetche[66] · John Apostolakis[5] · Makoto Asai[63] · Luca Atzori[5] · Marian Babik[5] · Giuseppe Bagliesi[32] · Marilena Bandieramonte[5] · Sunanda Banerjee[16] · Martin Barisits[5] · Lothar A. T. Bauerdick[16] · Stefano Belforte[35] · Douglas Benjamin[82] · Catrin Bernius[63] · Wahid Bhimji[46] · Riccardo Maria Bianchi[105] · Ian Bird[5] · Catherine Biscarat[52] · Jakob Blomer[5] · Kenneth Bloom[97] · Tommaso Boccali[32] · Brian Bockelman[97] · Tomasz Bold[43] · Daniele Bonacorsi[25] · Antonio Boveia[101] · Concezio Bozzi[28] · Marko Bracko[41,93] · David Britton[86] · Andy Buckley[86] · Predrag Buncic[5] · Paolo Calafiura[46] · Simone Campana[5] · Philippe Canal[16] · Luca Canali[5] · Gianpaolo Carlino[31] · Nuno Castro[47,96] · Marco Cattaneo[5] · Gianluca Cerminara[5] · Javier Cervantes Villanueva[5] · Philip Chang[75] · John Chapman[76] · Gang Chen[23] · Taylor Childers[1] · Peter Clarke[83] · Marco Clemencic[5] · Eric Cogneras[50] · Jeremy Coles[76] · Ian Collier[61] · David Coll Gloria Corti[5] · Gabriele Cosmo[5] · Davide Costanzo[112] · Ben Couturier[5] · Kyle Cranmer[57] · Jack Cranshaw[1] · Leonardo Cristella[26] · David Crooks[86] · Sabine Crépé-Renaudin[52] · Robert Currie[83] · Sünje Dallmeier-Tiessen[5] ·

Machine Learning in High Energy Physics Community White Paper

May 17, 2019

Abstract: Machine learning has been applied to several problems in particle physics research, beginning with applications to high-level physics analysis in the 1990s and 2000s, followed by an explosion of applications in particle and event identification and reconstruction in the 2010s. In this document we discuss promising future research and development areas for machine learning in particle physics. We detail a roadmap for their implementation, software and hardware resource requirements, collaborative initiatives with the data science community, academia and industry, and training the particle physics community in data science. The main objective of the document is to connect and motivate these areas of research and development with the physics drivers of the High-Luminosity Large Hadron Collider and future neutrino experiments and identify the resource needs for their implementation. Additionally we identify areas where collaboration with external communities will be of great benefit.

How the MLaaS4HEP training workflow works

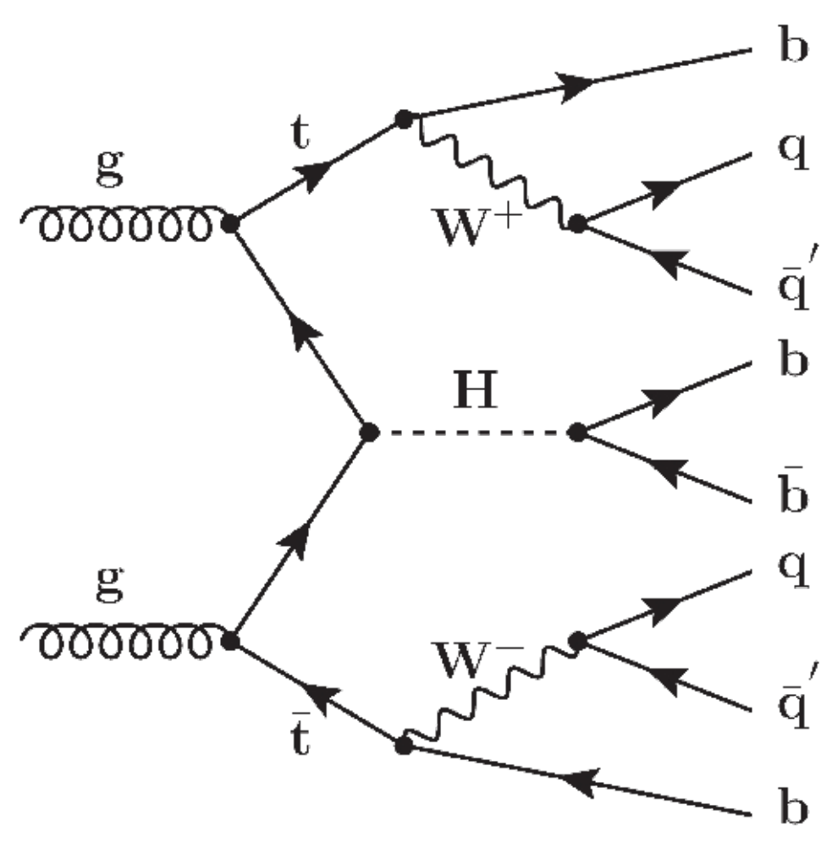

1
Read all the ROOT files
uproot
B
ROOT
compute specs file
specs.json
{max:
{key1: max_1,
key_2: max_2,...},
min:
{key_1: min_1, key_2:
min_2,...}, ...}
load specs information
2
list of files
If chunk c_i is empty or fully processed, read N_{chunk} events from the file f_i
uproot
S
B
F
ROOT
i = 0
i = i + 1
read the events
NO
YES
Are all the files completely read?
Take $N_{chunk} \cdot n_i / N_{tot}$ events from the chunk c_i
convert into numpy arrays, fix Jagged Arrays' dimension and normalise the values
chunk of handled events
jagged branches
flat branches
padding
jagged branch
rest of jagged branches with padding
data array
Did you go through all the files?
NO
YES
pre-process the events
Train the model for N_{epochs} using batches of data with size N_{batch}
K
train the ML model

The preprocessing operations are applied both during the reading phase and during the preprocessing of the events

Validation and performance test of MLaaS4HEP

We chose a signal vs background discrimination problem in a $t\bar{t}H$ analysis as a real physics use-case to:

1. validate the MLaaS4HEP results from the physics point of view
2. test the performance of MLaaS4HEP framework



We used a simple Neural Network in Keras obtaining an AUC score comparable with the BDT-based analysis, performed within the TMVA framework by a subgroup of the CMS HIG PAG.

To test the performance we used a dataset (without any physics cut) with 28.5M events and a total size of about 10.1 GB. The tests were performed using two different resources[1] and the ROOT files were read from local file-systems (SSD storages) and remotely from Grid sites.

- specs computing phase (chunk size = 100k events)
  - Event throughput: 8.4k – 13.7k evts/s
  - Total time using all the 28.5M events: 35 – 57 min
- chunks creation in the training phase (chunk size = 100k events)
  - Event throughput: 1.1k – 1.2k evts/s
  - Total time using all the 28.5M events: 6.5 – 7.5 hrs

No preprocessing operations were applied within MLaaS4HEP in these tests

[1] macOS, 2.2 GHz Intel Core i7 dual-core, 8 GB of RAM and CentOS 7 Linux, 4 VCPU Intel Core Processor Haswell 2.4 GHz, 7.3 GB of RAM CERN Virtual Machine

Creation of the MLaaS4HEP cloud service

The goal is to create a **cloud service** that could use cloud resources and could be added into the INFN Cloud portfolio of services. To provide this we worked through a series of steps.

- We built a **MLaaS4HEP server** with some APIs using the (Python-based) Flask framework
- We integrated an **OAuth2 Proxy server** to manage users authentication/authorization
- We integrated an **XRootD Proxy server** to allow the usage of an X.509 proxy for the remote access to ROOT data
- We connected the MLaaS4HEP server with **TFaaS** in a way that the ML models trained by the MLaaS4HEP server are saved in a repository from which the TFaaS service can take them for the inference phase

We implemented a working prototype [4] connecting the aforementioned services hosted by a VM of the INFN Cloud. The MLaaS4HEP server APIs can be reached at the following address https://90.147.174.27:4433 while TFaaS at https://90.147.174.27:8081.

Once the user obtains an access token from the authorization server, he/she can contact the MLaaS4HEP server or TFaaS as in the following ways:

```
curl -L -k -H "Authorization: Bearer ${TOKEN_MLAAS}" -H "Content-Type:
application/json" -d @submit.json https://90.147.174.27:4433/submit
```

Submit a MLaaS4HEP training workflow

```
curl -L -k -H "Authorization: Bearer ${TOKEN_TFAAS}" -X POST -H "Content-type:
application/json" -d @predict_bkg.json https://90.147.174.27:8081/json
```

Get prediction from TFaaS

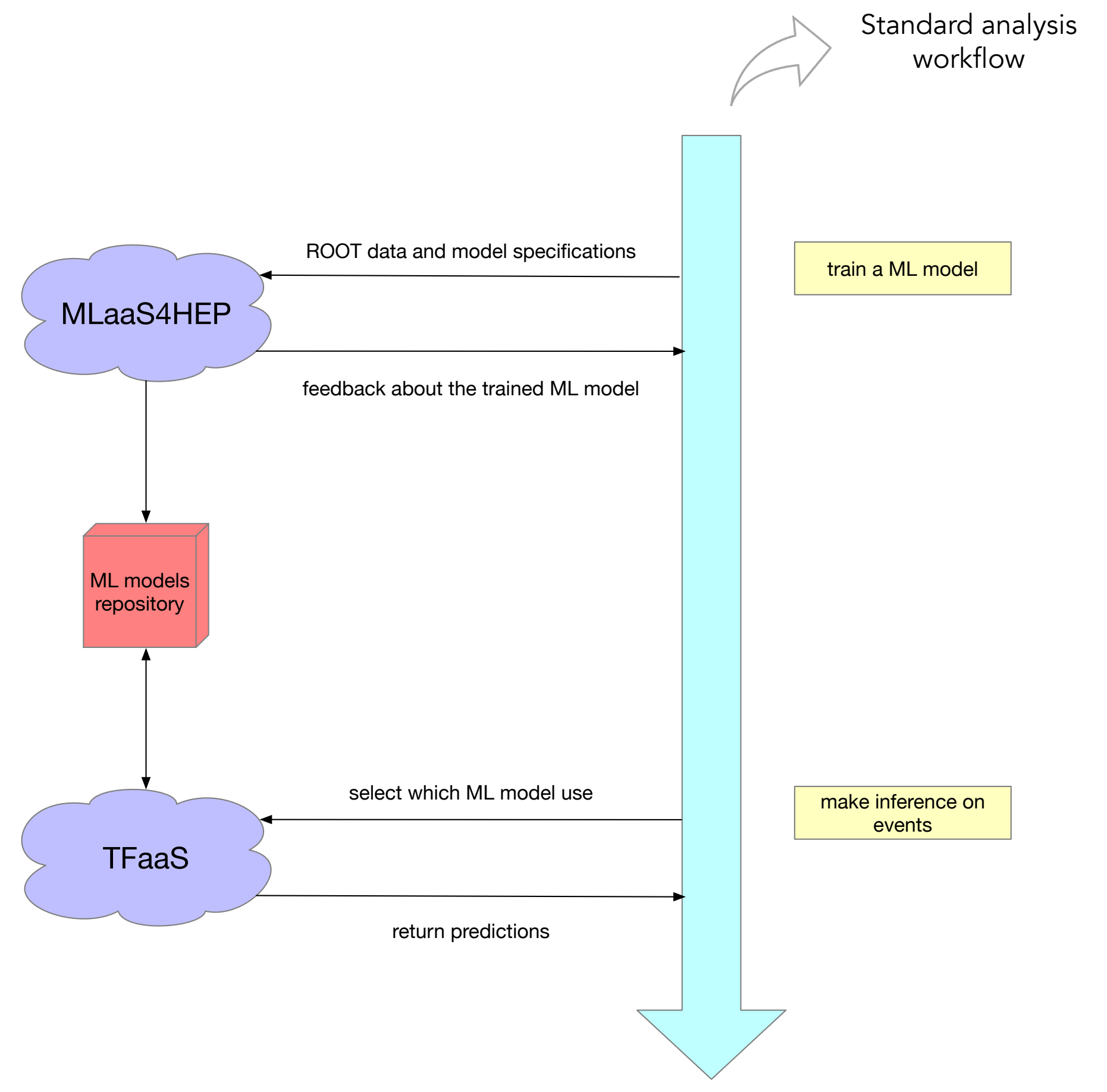
How the analysts can use the service

The MLaaS4HEP framework has a multi-language architecture (Python and Go) and it is structured in three layers:

- **Data Streaming Layer** is responsible for local and remote data access of HEP ROOT files (based on the uproot library)
- **Data Training Layer** is responsible for feeding HEP ROOT data into the ML framework of user choice
- **Data Inference Layer** (TFaaS) provides access to pre-trained TF models via HTTP protocol

The first two layers contributes to the **MLaaS4HEP** [2] training workflow, while **TFaaS** [3] is encharged of the inference phase.

Let's suppose that a CMS analyst performs an analysis workflow and at a certain point he/she wants to train a ML model to make predictions on some events. The analyst has the possibility to contact the MLaaS4HEP service, providing information about the ROOT data and the ML model/framework to use, to train the ML model which is subsequently stored in a repository from which the TFaaS service has access. Afterwards, the analyst contact the TFaaS service specifying which of the pre-trained ML models use to obtain predictions for the events.


Standard analysis workflow
ROOT data and model specifications
MLaaS4HEP
feedback about the trained ML model
train a ML model
ML models repository
select which ML model use
TFaaS
return predictions
make inference on events

Run the MLaaS4HEP training workflow

```
./workflow.py --files=files.txt --labels=labels.txt --model=model.py --params=params.json --preproc=preproc.json
```

MLaaS4HEP training workflow

Input ROOT files

Labels related to ROOT files content

Definition of the ML model

MLaaS4HEP parameters

Definition of preprocessing operations

files.txt

```
<PATH>/ttH_signal.root
<PATH>/TTBar.root
<PATH>/QCD_HT1000to1500.root
```

labels.txt

```
1
0
0
```

model.py

```
from tensorflow import keras

def model(idim):
ml_model = keras.Sequential([
              keras.layers.Dense(128, activation='relu', input_shape=(idim,),
name='inputs'),
              keras.layers.Dropout(0.5),
              keras.layers.Dense(64, activation='relu'),
              keras.layers.Dropout(0.5),
              keras.layers.Dense(1, activation='sigmoid')])
ml_model.compile(optimizer=keras.optimizers.Adam(lr=1e-3),
                 loss=keras.losses.BinaryCrossentropy(),
                 metrics=[keras.metrics.BinaryAccuracy(name='accuracy'),
                 keras.metrics.AUC(name='auc')])
return ml_model
```

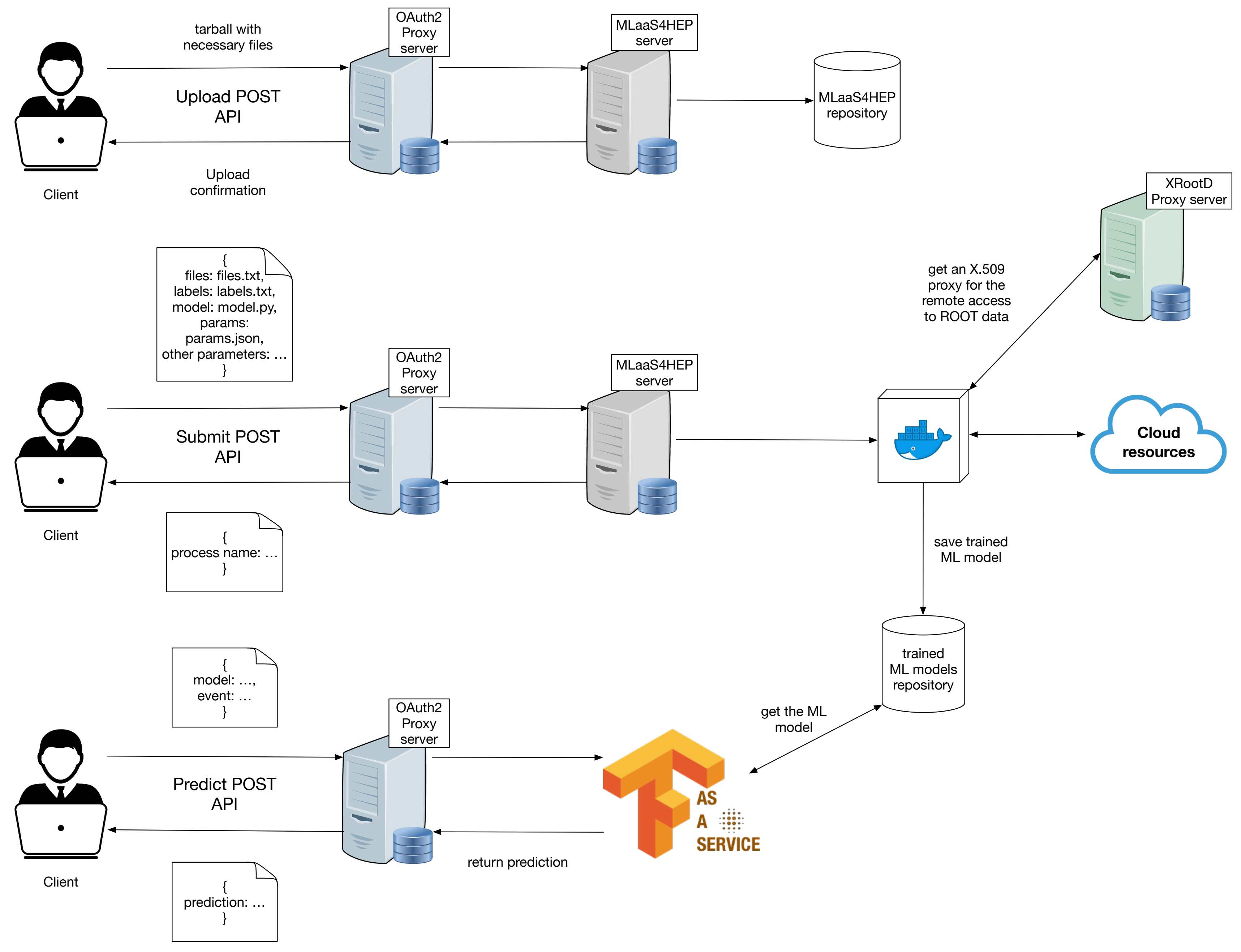
params.json

```
{
"nevts": 30000,
"chunk_size": 10000,
"epochs": 5,
"batch_size": 100,
"branch": "boostedAk8/events",
"selected_branches":"",
"exclude_branches": "",
"redirector":
"root://stormgf1.pi.infn.it",
}
```

preproc.json

```
{"new_branch": {

   "log_partonE": {
     "def": "log(partonE)",
     "type": "jagged",
     "cut_1": ["log_partonE<6.31", "any"],
     "cut_2": ["log_partonE>5.85", "all"],
     "remove": "False",
     "keys_to_remove": ["partonE"]},

   "nJets_square": {
     "def": "nJets**2",
     "type": "flat",
     "cut": "1<=nJets_square<=16",
     "remove": "False",
     "keys_to_remove": ["nJets"]}},

 "flat_cut": {
   "nLeptons": {
   "cut": "0<=nLeptons<=2",
   "remove": "False"}},

 "jagged_cut": {
   "partonPt": {
   "cut": ["partonPt>200", "all"],
   "remove": "False"}}}
```

Working prototype at INFN Cloud



Conclusions

MLaaS4HEP is an R&D project started in 2018 within CMS. During these years MLaaS4HEP was validated, tested and improved adding new features. Recently we worked towards a cloud service that can be integrated in the INFN Cloud portfolio of services. In this work we presented a working prototype of such service which allows HEP users to submit ML training workflows and get predictions simply using HTTPS calls.

References

[1] V. Kuznetsov, L. Giommi, D. Bonacorsi, MLaaS4HEP: Machine Learning as a Service for HEP. Comput Softw Big Sci 5, 17 (2021). DOI: 10.1007/s41781-021-00061-3
[2] Machine Learning as a Service for HEP (MLaaS4HEP). URL: https://github.com/vkuznet/MLaaS4HEP
[3] Tensorflow as a Service (TFaaS). URL: https://github.com/vkuznet/TFaaS
[4] MLaaS4HEP server and working prototype. URL: https://github.com/lgiommi/MLaaS4HEP_server

Contact

luca.giommi3@unibo.it

ICHEP 2022 BOLOGNA