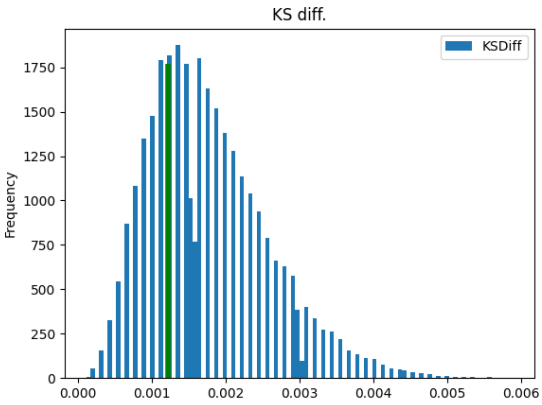
ValidationsTools

Table des matières

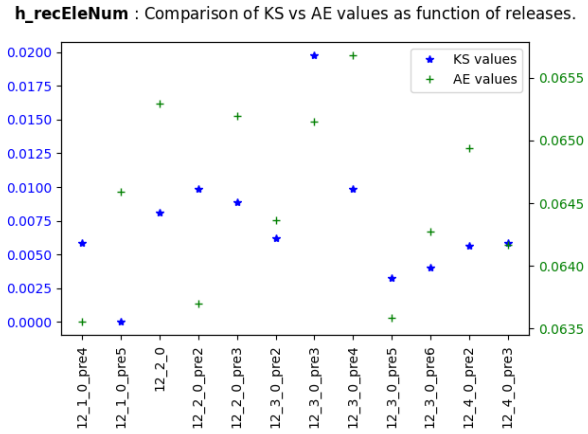
General explanation :

This repository get all the validations tools such as Kolmogorov-Smirnov (KS Tools) or AutoEncoders Tools (AE).

KS : talk about first dev (1 release vs 1 reference) to (1 reference vs lot of releases). pValues.
From the values of an histogram, we can construct an integrated cumulative curve. From this curve we can take the maximum of the difference between 2 consecutives values. Repeating this operation with a lot of ROOT files (200, 500 or 1000) we can construct a KS curve such here onto the right.
From this curve we can extract a pValue and try to obtain an idea of the validity of the histo and then the release.



AE : analyze on a reference and comparison with a lot of releases. The precedent explanation was made for one release vs 1 reference. Keeping the reference we can compare with a lot of releases (here about 12).
From the ROOT files presented above we can train an autoencoder and try to predict the result of a given entry for one release. Doing this for a tenth of releases we can obtain a comparison, function of the releases, for the pValues or the differences at the end of the AE.
This is shown onto the left here, and extract an idea about the validity of one (or more) release(s).



One goal of those operations is from a reference release, to create a lot of ROOT files from this reference release. The set of the reference ROOT files are named as RRF or created reference ROOT Files (CRRF).

Once you have created all those files (see 6), you will need to think about comparison.
In KS you have a reference release, and one or more release(s) to compare with. Into the first version of the KS tool, we have one « official » reference release (and a lot of created ROOT files from the same reference release). From those files we can compare a new release (in general called rel by comparison with the ref file) with the reference release and extract a lot of pictures for each histogram.

KS curves

First we generate an average curve bin to bin from the mean of all the histograms from RRF.
For each histogram we generate a graph with 3 curves representing the classical comparison of the histograms (ref vs rel) with the average curve added.
The new histo curve is in red, the reference one is in blue and the average curve in green.

We have 3 graphs more. The first one represents the cumulatives curves of the 3 curves described above. A second graph representing the difference curve between the 2 (new/rel and reference/ref) cumulatives curves. The last picture represents the difference curve between the new curve and the average curve.

With those 4 preceding pictures, we add 3 new pictures names as KS pictures and representing the Kolmogorov curves (see first picture above). For the first graph, we use all the references histograms and compare each cumulated curve with all the others.
For the second graph, we choose a random histogram as reference and compare its cumulated curve with each cumulated curve of all the others.
For the third graph, we took the histogram to be validated and compare its cumulated curve with each other cumulated curve of all the others histograms.
The last three curves are created with KS_Tools while the fourth others are created with a simple validation routine.

AE curves

For AE, we have a lot of difficulties. instead of KS, we have to search for a correct AE (number of layers, number of points for those layers, …) and generate some pictures for a lot of releases (those placed into the DATA folder).
We can generate some comparisons with KS results (such as into the second picture above) as a function of a release, generate a prediction for each histo, and for each release. This can give some advice for the validity of the histogram and then for a release.

All thoses files are located into a folder named ValidationsTools which contains at least 5 folders :
- Doc : containing this documentation,
- ZEE_Flow : folder used for the creation of the ROOT files,
- KS_Tools : folder used for the KS pictures and files,
- AutoEncoder : containing the autoencoder tools and results,
- DATA : empty at the installation. Contain the « official » ROOT files, i.e. the releases ROOT files used for the comparison mentioned above.

⚠later create a link to an explanation of the common files used for the ROOT files creation (CommonFiles/path...).
- CommonFiles : with the files which are used by all the tools, KS as AE.

Here is a summary of all the files/folders used for this tool.

CommonFiles : files used by all scripts

default.py
pathsLLR.py
pathsLocal.py
pathsPBS.py
rootValues.py
sources.py
defaultStd.py
defaultDiff.py

Doc : containing the documentation

DATA : empty at the installation. Contain the « official » ROOT files.

RESULTFOLDER : empty at the installation. Contain the «created» ROOT files (CRF).

ZEE_Flow : folder used for the creation of the ROOT files

step1.py
step2.py
step3.py
step4.py
zee_flow.sh
zee_reduce.sh
extractValues.sh
reduceSize1File.py : reduce the size of the created ROOT files

KS_Tools : folder used for the KS pictures and files

extractValues.py : create a file per histo with all values from the CRF
createFiles.py : create some pictures for KS
createFiles_v2.py : create a pValues file and a diffmax file for AE

AutoEncoder : containing the autoencoder tools and results

lossValuesVsKS.py
pytGrosDiff-v01_lite.py
pytGros-v03.py
pytGros-v02_lite.py
pytGrosLoop-v03.py
zeeNewFiles.py
zeeKSvsAEComp.py
zeeKSComp.py
zeeMapDiff2.py
zeepValues.py
DATASETS : initially contain DATA files
RESULTS : contain the results of AE

Launching the installation :

- git clone https://github.com/archiron/ValidationsTools ValidationsTools
- cd ValidationsTools
- chmod 755 *.sh

then launch : . installValidationsTools.sh

This install the library (ChiLib), the release file env (cmsrel $Release $Release) and copy the ZEE_Flow/CCA or ZEE_Flow/LLR files (depending of the site you are working on - the CC or the LLR site) onto the $Release/src/Kolmogorov folder.
$Release is the release you want working with.

Launching the ROOT files creation :

From the top of the Tools (i.e. ValidationsTools), then launch :
. createROOTFiles.sh
This script launch the creation of the ROOT files, using the step[1-4].py scripts.
You can use own but they have to be similar to the existing ones (see 8).

!! talk about the minimum size needed for nb_evts (~ 1000). 10 is too small.

Reducing size :

When ALL ROOT files are created, launch (always from the top folder) :
. reduceROOTSize.sh
this will reduce the size of the ROOT files (typically from 150/200 Mo to 1.5/2 Mo), keeping the name of the file.

Extracting values :

When all the ROOT files are created and reduced, we need to creates 1 file per histo with all the histo values for each ROOT file.
It is the first job of the extractValues[_init].sh scripts.

It can be launched with : . extractValues_init.sh
and create into the RESULTFOLDER (see 9) a lot of text files. All ROOT files are read, and then for each histo, the values of the histo curve is stored into an array, ROOT file after ROOT file.

Once we have all the ROOT files read, the array is stored into a text file dedicated to this histo.
The name is : RESULTFOLDER + 'histo_' + str(leaf) + '_' + '{:03d}'.format(nbFiles) + '.txt' where leaf is the name of the histo, nbFiles the number of the ROOT files (same as NB_EVTS in rootValues.py – see 9).

Example :

histo_h_ele_zEff_200.txt
histo_h_ele_seedDphi2_200.txt

Reference, comparison files :

Talk about the files to add into the DATA folder.

Checking ROOT files :

When ALL ROOT files are reduced, launch (always from the top folder) :
. checkRootFiles_init.sh

this will make a test of the ROOT files (those which are generated but also those which are added), and have an idea of some pbm which can occur. The pbm are generally a pbm of number of histograms.

ANNEXE

In that follow, we have the following notation :

al numeric values (int/float) are in **blue**.

Annexe : ROOT scripts to modify

Here we are talking about the step[1-4].py scripts.

In all stepX.py files, we need to replace the 2 lines :

```
from Configuration.Eras.Era_Run3_cff import Run3

process = cms.Process('SIM',Run3)
```

with :

```
from Configuration.Eras.Era_Run3_cff import Run3

if len(sys.argv) > 1:
print(sys.argv)
print("step 1 - arg. 0 :", sys.argv[0])
print("step 1 - arg. 1 :", sys.argv[1])
print("step 1 - arg. 2 :", sys.argv[2])
print("step 1 - arg. 3 :", sys.argv[3])
print("step 1 - arg. 4 :", sys.argv[4])
ind = int(sys.argv[2])
max_number = int(sys.argv[4])
else:
print("step 1 - rien")
ind = 0
path = ''
max_number = 10 # number of events

max_skipped = ind * max_number

process = cms.Process('SIM',Run3)
```

And also replace :

```
fileName = cms.untracked.string('file:step1.root'),
```

with :

```
fileName = cms.untracked.string('file:step1_' + '%0004d'%max_number + '_' +
'%003d'%ind + '.root'),
```

And for all file name such as step2.root or other.

In the last script (usually step4.py) which generate the DQM file, we have to insert 2 lines between the dqmsave_step line such as :

```
part3 = '/RECO_' + '%0004d'%max_number + '_' + '%003d'%ind
process.dqmSaver.workflow = '/Global/' + 'CMSSW_X_Y_Z' + part3
process.dqmsave_step = cms.Path(process.DQMSaver)
```

Annexe : paths definitions

RESULTFOLDER : path where you want the created ROOT files are located. It also contain the text files for each histo.

Annexe : values for ROOT creation

For the creation of the ROOT files we need python files with physics and for those files we need some values.
In general we need a lot files and not only one. So, those file went from **Nbegin = 0** to **Nend=200** or **1000**.
For a lot of tests it can be nice to have smaller values such as :
Nbegin = 20
Nend = 23

Inside the python files we nned to know about the number of events to be used. Most of the « official » runs use **NB_EVTS = 9000**. For some rapid tests we can use :
NB_EVTS = 10
but for tests with values expected, we need NB_EVTS = 1000.