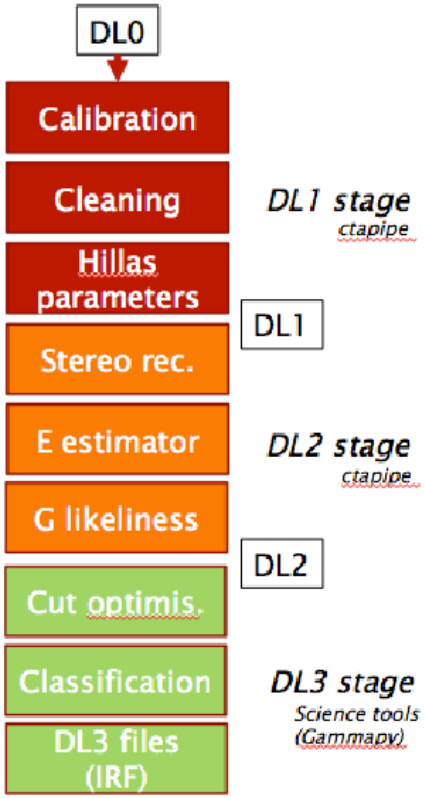
Protopipe

Alice Donini

CTA INFN Meeting - 1 July 2020

Protopipe

- Public pipeline prototype for CTA from DL0 to IRF
- Based on latest stable ctapipe release
- The pipeline provides scripts to:
  - Process simtelarray files and write DL1 or DL2 tables
  - Build regression or classification models with diagnostic plots
  - Estimate the best cutoffs which gives the minimal sensitivy reachable in a given amount of time
  - Produce instrument response functions (IRF), including sensitivity
- Still under development

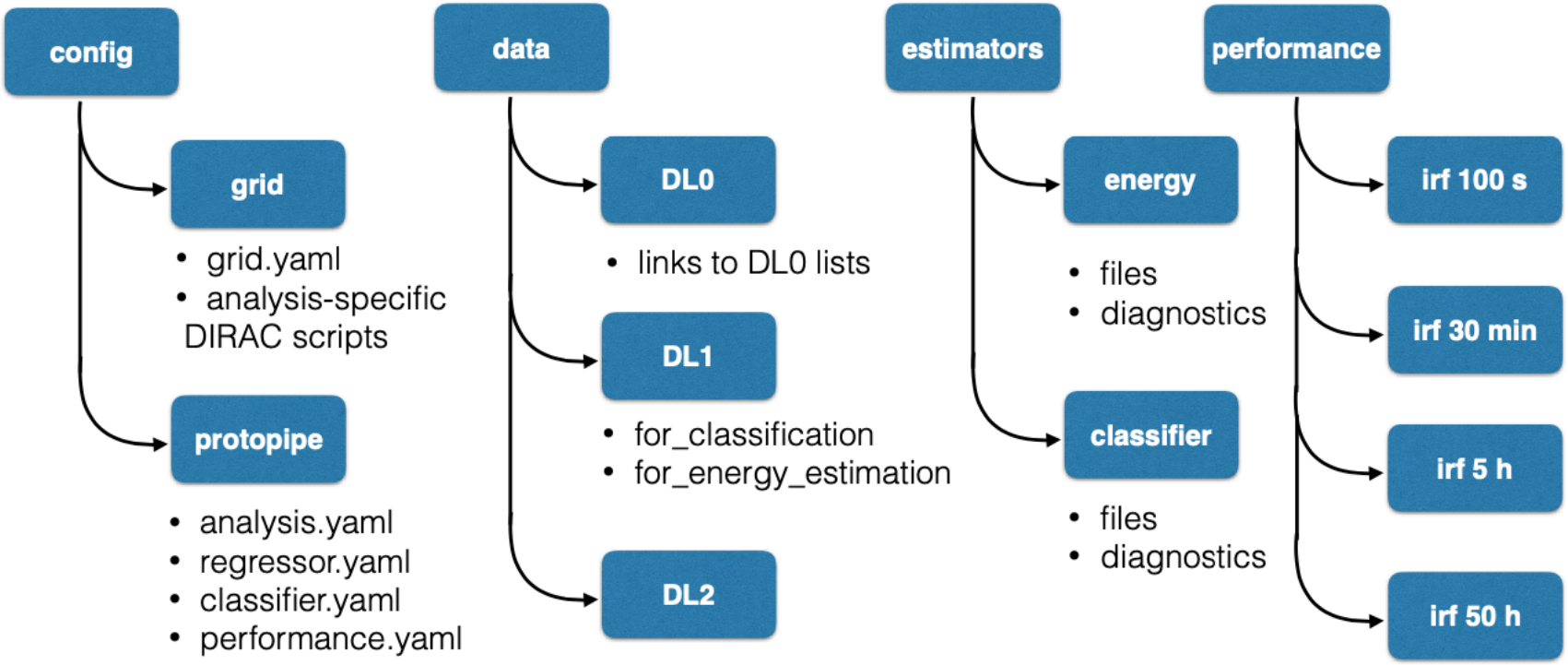


Components from ctapipe

- Calibration
  - charge and pulse times extraction via *ctapipe.image.extractors. TwoPassWindowSum*
- Image cleaning
  - performed using *ctapipe.image.cleaning. mars_cleaning_1st_pass*
  - the settings are user-dependent
- Parametrization
  - performed by *ctapipe.image.hillas. hillas_parameters*
- Direction reconstruction
  - performed via *ctapipe.reco.HillasReconstructor* with a minimum number of 2 surviving images per event.

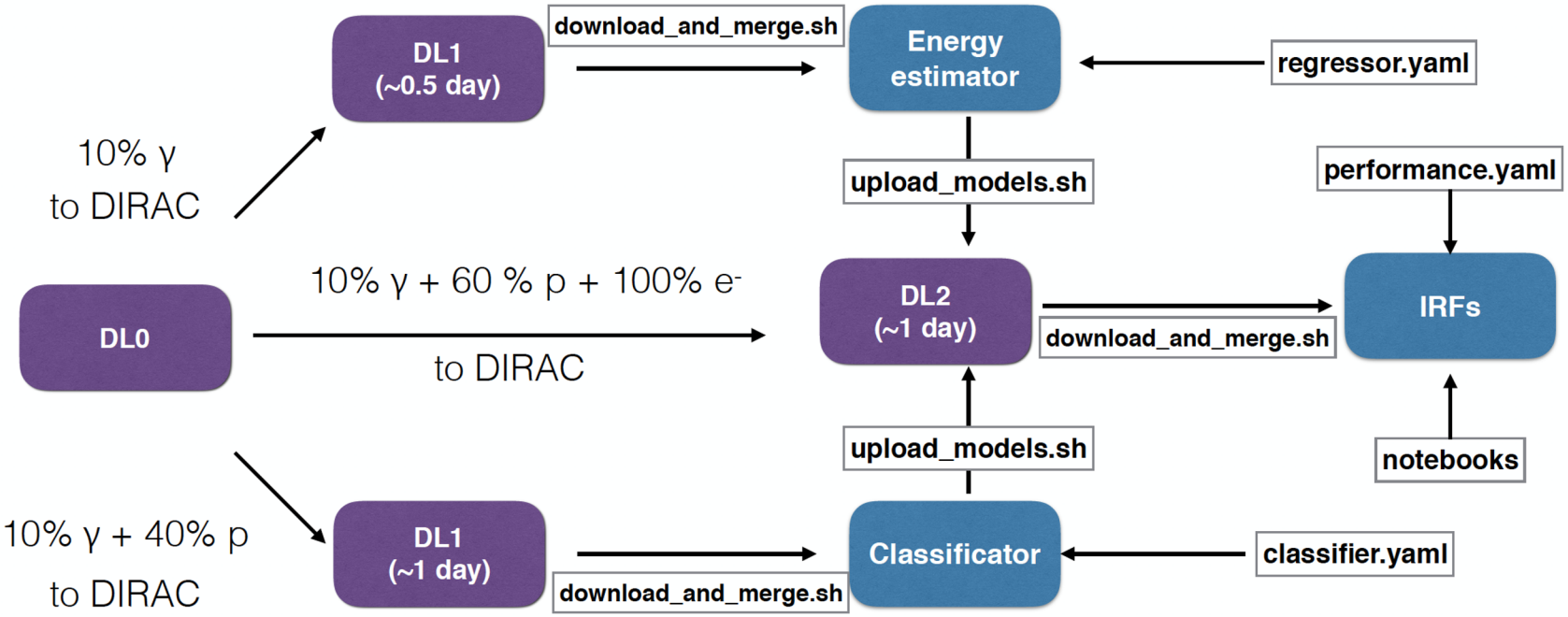
Local folder

create an analysis parent folder with the auxiliary script create_dir_structure.py



Workflow

In the following : '*to DIRAC*' requires '*grid.yaml*' and '*analysis.yaml*'



M. Peresano talk in Lugano: https://indico.cta-observatory.org/event/1995/

Energy estimator & Gamma hadron classifier

1. Data training: tables of image and stereo parameters that will be further used to build energy estimators

```
usage: write_dl1.py [-h] --config_file CONFIG_FILE -o OUTFILE [-m MAX_EVENTS]
                    [-i INDIR] [-f [INFILE_LIST [INFILE_LIST ...]]]
                    [--wave_dir WAVE_DIR] [--wave_temp_dir WAVE_TEMP_DIR]
                    [--wave | --tail] [--save_images] [--estimate_energy]
                    [--regressor_dir REGRESSOR_DIR]
```

2. Model building: estimation models for particle energy and gamma/hadron classification

```
>$ ./build_model.py --help
usage: build_model.py [-h] --config_file CONFIG_FILE [--max_events MAX_EVENTS]
                      [--wave | --tail]
```

Regressor.yaml

```yaml
General:
 model_type: 'regressor'
 # [...] = your analysis local path
 # Please, refer to directory structure shown at Lugano
 data_dir: '[...]/data/DL1/for_energy_estimation'
 data_file: 'dl1_{}_gamma_merged.h5'
 outdir: '[...]/estimators/energy_regressor'
 cam_id_list: ['LSTCam', 'NectarCam']
 table_name_template: 'feature_events_'

Split:
 train_fraction: 0.8

Method:
 name: 'AdaBoostRegressor'
 target_name: 'mc_energy'
 tuned_parameters:
  learning_rate: [0.3]
  n_estimators: [100]
  base_estimator__max_depth: [null]  # null is equivalent to None
  base_estimator__min_samples_split: [2]
  base_estimator__min_samples_leaf: [10]
 scoring: 'explained_variance'
 cv: 2
```

```yaml
FeatureList:
 - 'log10_charge'
 - 'log10_impact'
 - 'width'
 - 'length'
 - 'h_max'

SigFiducialCuts:
 - 'xi <= 0.5'

Diagnostic:
 # Energy binning (used for reco and true energy)
 energy:
  nbins: 15
  min: 0.0125
  max: 125
```

Classifier.yaml

```yaml
General:
 model_type: 'classifier'
 # [...] = your analysis local path
 # Please, refer to directory structure shown at Lugano
 data_dir: '[...]/data/DL1/for_classification/'
 data_sig_file: 'dl1_tail_gamma_merged.h5'
 data_bkg_file: 'dl1_tail_proton_merged.h5'
 cam_id_list: ['LSTCam', 'NectarCam']
 table_name_template: 'feature_events_'  # Will be completed with cam_ids
 outdir: '[...]/estimators/gamma_hadron_classifier'

Split:
 train_fraction: 0.8
 use_same_number_of_sig_and_bkg_for_training: False  # Lowest statistics will drive the split

Method:
 name: 'RandomForestClassifier'  # AdaBoostClassifier or RandomForestClassifier
 target_name: 'label'
 tuned_parameters:
  n_estimators: [200]
  max_depth: [10]  # null for None
  min_samples_split: [10]
  min_samples_leaf: [10]
 scoring: 'roc_auc'
 cv: 2
 use_proba: True  # If not output is score
 calibrate_output: False  # If true calibrate probability
```

```yaml
FeatureList:
 - 'log10_reco_energy'
 - 'width'
 - 'length'
 - 'skewness'
 - 'kurtosis'
 - 'h_max'

SigFiducialCuts:
 - 'offset <= 0.5'

BkgFiducialCuts:
 - 'offset <= 1.'

Diagnostic:
 # Energy binning (used for reco and true energy)
 energy:
  nbins: 4
  min: 0.0125
  max: 125
```

DL2 production

- produce tables of gamma-rays, hadrons and electrons with event informations

```
usage: write_dl2.py [-h] --config_file CONFIG_FILE -o OUTFILE [-m MAX_EVENTS]
                    [-i INDIR] [-f [INFILE_LIST [INFILE_LIST ...]]]
                    [--wave_dir WAVE_DIR] [--wave_temp_dir WAVE_TEMP_DIR]
                    [--wave | --tail] [--regressor_dir REGRESSOR_DIR]
                    [--classifier_dir CLASSIFIER_DIR]
                    [--force_tailcut_for_extended_cleaning FORCE_TAILCUT_FOR_EXTENDED_CLEANING]
                    [--save_images]
```

Perfomance.yaml

```yaml
analysis:

 # Theta square cut optimisation (opti, fixed, r68)
 thsq_opt:
  type: 'opti'
  value: 0.2  # In degree, necessary for type fixed

 # Normalisation between ON and OFF regions
 alpha: 0.2

 # Minimimal significance
 min_sigma: 5

 # Minimal number of gamma-ray-like
 min_excess: 10

 # Minimal fraction of background events for excess comparison
 bkg_syst: 0.05

 # Reco energy binning
 ereco_binning:  # TeV
  emin: 0.012589254
  emax: 199.52623
  nbin: 21

 # Reco energy binning
 etrue_binning:  # TeV
  emin: 0.019952623
  emax: 199.52623
  nbin: 42
```

```yaml
particle_information:
 gamma:
  n_events_per_file: 1000000  #  10**5 * 10
  e_min: 0.003
  e_max: 330
  gen_radius: 1400
  diff_cone: 0
  gen_gamma: 2

 proton:
  n_events_per_file: 4000000  #  2 * 10**5 * 20
  e_min: 0.004
  e_max: 600
  gen_radius: 1900
  diff_cone: 10
  gen_gamma: 2
  offset_cut: 1.

 electron:
  n_events_per_file: 2000000  #  10**5 * 20
  e_min: 0.003
  e_max: 330
  gen_radius: 1900
  diff_cone: 10
  gen_gamma: 2
  offset_cut: 1.

column_definition:
 # Column name for true energy
 mc_energy: 'mc_energy'
 # Column name for reconstructed energy
 reco_energy: 'reco_energy'
 # Column name for classification output
 classification_output:
  name: 'gammaness'
  range: [0, 1]
 angular_distance_to_the_src: 'xi'
```

Optimized cuts and IRFs

- estimate performance

```
usage: make_performance.py [-h] --config_file CONFIG_FILE --obs_time OBS_TIME
                           [--wave | --tail]
```

- find the best cutoff in gammaness/score, to discriminate between signal and background, as well as the angular cut to obtain the best sensitivity for a given amount of observation time and a given template for the source of interest
- compute the instrument response functions, effective area, point spread function and energy resolution
- estimate the sensitivity

Latest updates

- Benchmark notebook for comparison between protopipe and CTA-MARS
  - DL1
    - calibration | *benchmarks_DL1_calibration.ipynb*
    - Image cleaning| *benchmarks_DL1_image-cleaning.ipynb*
  - DL2
    - Direction reconstruction| *benchmarks_DL2_direction-reconstruction.ipynb*
    - Energy estimation| *benchmarks_DL2_energy-estimation.ipynb*
    - Particle classification| *benchmarks_DL2_particle_classification.ipynb*
  - DL3
    - Point Spread Function| *benchmarks_DL3_PSF.ipynb*
    - Instrument Response Function and sensitivity| *benchmarks_DL3_IRFs_and_sensitivity.ipynb*

- GRID support
  - interface to the GRID is needed

- New IRF builder tool
  - https://github.com/cta-observatory/pyirf

- New release based on ctapipe 0.8 soon