Online Analysis of High-Volume Data Streams in Astroparticle Physics

Christian Bockermann[1], Kai Brügge[2], Jens Buss[2], Alexey Egorov[1], Katharina Morik[1], Wolfgang Rhode[2], and Tim Ruhe[2]

[1] Computer Science Department
Technische Universität Dortmund
44221 Dortmund, Germany
[2] Astroparticle Physics Department
Technische Universität Dortmund
44221 Dortmund, Germany
firstname.lastname@udo.edu

Abstract. Experiments in high-energy astroparticle physics produce large amounts of data as continuous high-volume streams. Gaining insights from the observed data poses a number of challenges to data analysis at various steps in the analysis chain of the experiments. Machine learning methods have already cleaved their way selectively at some particular stages of the overall data mangling process.
In this paper we investigate the deployment of machine learning methods at various stages of the data analysis chain in a gamma-ray astronomy experiment. Aiming at online and real-time performance, we build up on prominent software libraries and discuss the complete cycle of data processing from raw-data capturing to high-level classification using a data-flow based rapid-prototyping environment. In the context of a gamma-ray experiment, we review user requirements in this interdisciplinary setting and demonstrate the applicability of our approach in a real-world setting to provide results from high-volume data streams in real-time performance.

Keywords: online analysis, high-volume streams, astroparticle physics

1 Introduction

Modern astronomy studies celestial objects (stars, nebulae or active galactic nuclei) partly by observing high-energy beams emitted by these sources. By a spectral analysis of their emissions, these objects can be characterized and further insight can be derived. Plotting the energy emissions over time leads to a *light curve*, which may show pulsatile behavior and other properties that allow for a classification of the observed object. An example is the distinction of different supernova types based on the form of their light curves [12]. The creation of a spectrum of the radiated energy levels is therefore a key skill. A collection of different monitoring techniques such as satellites [22], telescopes [21,2,18] or

water tanks [4,1] is deployed to observe different ranges of the electromagnetic radiation produced by the sources. A central problem in all these experiments is the distinction of the crucial gamma events from the background noise that is produced by hadronic rays and is inevitably recorded. This task is widely known as the *gamma-hadron* separation problem and is an essential step in the analysis chain. The challenge in the separation step is the high imbalance between signal (gamma rays) and background noise, ranging from 1:1000 up to 1:10000 and worse, which implies large amounts of data that need to be recorded for a well-founded analysis of a source. The high sampling rate and the growing resolution of telescope cameras further require careful consideration of scalability aspects when building a data analysis chain for scientific experiments.

In this work, we investigate the *online* use of classification methods for data filtering and spectrum creation in gamma-ray astronomy. We review the data flow of prominent experiments like MAGIC or FACT, and inspect the preprocessing chain from data acquisition to the extraction of a spectrum from a machine learning point of view. With respect to the scalability requirements we discuss the use of *distributed model* application as part of the analysis chain. As recorded data is unlabeled, only simulated data can be used to train filters. These can easily be trained in a batch setting, but an application of any model is required in an online fashion. This, in turn, poses constraints on the features that can be used. Further, there exists an *interdisciplinary gap* between the domain experts (physicists) and the field of computer science that spoils a fruitful cooperation in both areas. Our contributions are as follows:

(1) We outline the data flow from observation to analysis in the real-world setting of the FACT Cherenkov Telescope.
(2) We discuss multiple spots for the use of machine learning models in the analysis and the constraints faced.
(3) Based on the streams framework [9], we provide a library of processing functions (*fact-tools*) to easily model the overall data processing and analysis chain in a rapid prototyping manner.
(4) We demonstrate the applicability of our proposed framework and the machine learning methods using real-world data of the FACT telescope.

The rest of this paper is structured as follows: In Section 2 we provide a short overview over the field of Cherenkov astronomy, subject to our study, and review the flow of data from data acquisition to the extraction of the desired information. Along this data flow, we highlight the use of machine learning models, including related works on that matter, and close with a discussion on the requirements from the view of domain experts as well as the interdisciplinary gap that we faced between the world of end-users (physics) and data engineers (computer science). In Section 3 we introduce the *fact-tools* – our high-level framework to model the data flow, which integrates state of the art tools such as WEKA [16] and MOA [7] to incorporate machine learning for various tasks. We demonstrate the use of our framework and evaluate different machine learning methods in the real-world setting of the FACT telescope in Section 4. We summarize our *lessons learnt* and future ideas in Section 5.

2 Data Analysis in Cherenkov Astronomy

The examination of sources in astronomy relies on the observation of energy emitted by these sources. Unfortunately, these energy beams cannot be observed directly, but only by an indirect measuring of the effect they are triggering in some detector medium. In the case of Cherenkov telescopes, the atmosphere is used as detector medium: particles interact with elements in the atmosphere and induce cascading air showers as they pass the atmosphere. These showers emit so-called *Cherenkov light*, which can be measured by telescopes like MAGIC or FACT. Figure 1 shows an air shower triggered by some cosmic ray beam, emitting Cherenkov light that can be captured by the telescope camera. The cone of light produced by the shower is visible in the camera for a period of about 150 nanoseconds. The camera of the telescope consists of an array of light-sensitive pixels that allow for the recording of the light impulse induced by the air shower. For a fine-grained capture of the light pulses, the camera pixels are sampled at a very high rate (e.g. 2 GHz). Figure 1 shows the layout of the FACT camera, which consists of 1440 pixels in hexagonal form. The high sampling speed requires high-performance memory for buffering the sampled data. The cameras usually continuously sample all the pixels into a ring-buffer and a hardware trigger initiates a write-out to disk storage if some pixels exceed a specified threshold (i.e. indication of shower light hitting the telescope). Upon a trigger activation, the sampled data written to disk captures a series of camera samples which amount for a time period of about 150 to 300 nanoseconds, called the *region of interest* (ROI). This fixed-length series of consecutive camera samples produced upon a trigger is called an *event* and corresponds to the light cone induced by the airshower.

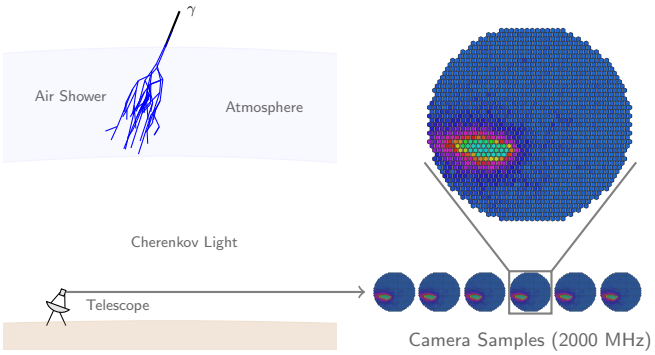

Fig. 1: An air shower produced by a particle beam hitting the atmosphere. The shower emits a cone of blue light (Cherenkov light) that will hit the telescope mirrors and is recorded in the camera. The right-hand side shows a still image of the light cone in the telescope camera (FACT telescope).

2.1 From Raw Data Acquisition to Spectral Analysis

The raw data produced by telescopes like MAGIC or FACT consists of the sampled voltages of the camera pixels for a given time period (ROI). Using these voltage levels, the following steps are required in the analysis, each of which is individually performed for each event, naturally implying a streamlined processing:

(1) *Calibration*, *Cleaning*: Calibrate the data, determine the pixels that are part of the light shower.
(2) *Feature Extraction*: Find features that best describe the data to solve the following steps.
(3) *Signal Separation*: Assess whether the event is induced by a gamma-ray (signal) or a hadronic shower (noise).
(4) *Energy Estimation*: Estimate the Energy of the primary gamma particle from the calculated energy correlated features.

Based on the number of signals detected and the estimated energy spectrum, properties of the observed astronomical source can be inferred. From a data analysis point of view we can map this process to the high-level data flow outlined in Figure 2. Especially the separation of signal and noise and the energy estimation are candidates for use of machine learning. The extraction of features for subsequent use of machine learning in steps (3) and (4) is a crucial step and requires back-to-back fine tuning with the learning methods. The dark arrows show additional back-to-back dependencies between the different steps. These dependencies induce a highly volatile optimization process in the development of the overall analysis chain: any changes in the calibration of cleaning may lead to slightly different feature values for the signal separation and the energy estimation step.

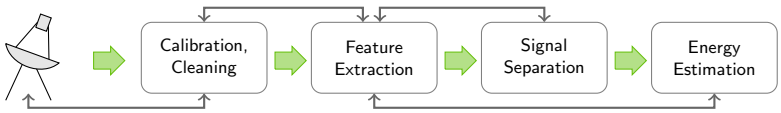

Fig. 2: Data processing steps from raw data acquisition to energy estimation.

The calibration and cleaning methods are highly domain specific and require careful consideration by domain experts. Given, that the electronics of such telescopes are customized prototypes, each device requires different setups that may vary with changes in the environment (temperature). These early steps in the data processing chain usually relate to hardware specifics and are fine-tuned in a manual way. In case of the FACT telescope, base voltages and gains for each pixels need to be adjusted with respect to calibration data recordings.

2.2 Feature Extraction for Machine Learning

Given domain knowledge, the gamma and hadronic particles behave different when hitting the atmosphere. As gamma particles are uncharged, they have strict directional energy and air showers created by gammas are expected to be directed straight from the source as well. Hadronic particles in contrast may be deviated by electromagnetic fields and thus will drop into the atmosphere from any direction. In addition, the atmospheric interaction of hadronic showers tends to degrade into much wilder cascades. A basic assumption is, that the structural differences in these showers are reflected in the image that the Cherenkov light emitted by these showers induces in the telescope camera. These properties are described by the so-called *Hillas parameters*, which form a set of geometric features that are widely used in gamma ray astronomy [19,13]. The features introduced by Hillas describe the orientation and size of an ellipse fitted to the area of a shower image. The ellipse is fitted to the pixels that have survived the previous *image cleaning* step, in which pixels not part of the shower are removed. The geometric orientation of the ellipse is correlated with the angular field of the telescope. Figure 3 shows a shower image after removal of non-shower pixel and the Hillas features derived. It is obvious, that the image cleaning step, in which the shower pixels are identified, has a direct impact on the ellipse, that will be fitted. Apart from the size (width, length) the orientation of the ellipse (alpha) and its offset from the origin is extracted.

In addition to these basic geometric features, other properties of shower images have been derived, such as the fluctual distribution from shower center [11] (for the Milagrito experiment) or the *surface brightness* [5]. In [14] Faleiro et al proposed the investigation of spectral statistics as discriminating features, whereas [23] evaluated an encoding of shower images using multi-fractal wavelets.

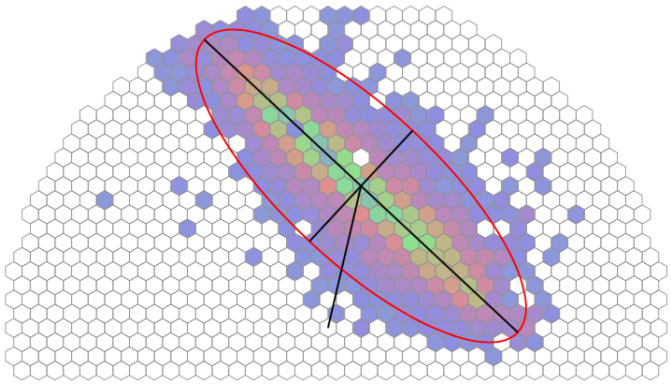

Fig. 3: Geometric *Hillas* features to support signal-noise separation. Figure shows an event in the FACT camera after image cleaning.

2.3 Signal Separation and Energy Estimation: Machine Learning

The detection of gamma induced events has been investigated as a binary classification task, widely referred to as the *gamma-hadron separation*. The features described in the previous section have all been proposed and tested in combination with a classification algorithm to achieve the best filtering of signal events from hadronic background. As the showers are distinct events, this boils down to finding some model

$$M : \mathcal{S} \to \{-1, +1\}$$

that maps a recorded shower $S \in \mathcal{S}$ represented by a set of features to one of the two possible classes. The challenge in this classification task is the highly imbalanced class distributions as a very small fraction of gamma rays needs to be separated from the large amount if showers induced by hadronic particles. The expected ratio is at the level of 1:1000 or worse, requiring a huge amount of recorded data in order to find a meaningful collection for training a classifier.

The classifiers tested with the aforementioned features range from manual threshold cuts using discriminative features [8], neural networks (combined with fractal features [23]) to support vector machines or decision trees. [8] provides a study comparing various classifiers on a fixed set of features (Hillas parameters). Random forests [10] generally provide a robust performance and have become a widely accepted method for the gamma-hadron separation in that domain.

Energy Estimation

Another field where machine learning contributes is the estimation of energy. The recorded data only reflects the image of light emitted by the air shower that was produced by the cosmic ray. Of interest to the physicists is the energy of the particles that induced the shower. The reconstruction of the energy of the primary particle can be seen as a *regression task*, finding a model

$$E : \mathcal{S} \to \mathbb{R}$$

which predicts the energy based on features obtained from the shower image. For the MAGIC telescope, Berger et al. investigated the energy reconstruction with random forests, claiming that a small set of features is suitable for a robust energy estimation, with the *size* parameter being the most important one [6].

Labeled Data by Probabilistic Simulation

A big problem when applying machine learning in astrophysics is, that particles arriving from outer space are inherently unlabeled. Using that data for supervised learning requires an additional step to obtain data for training a classifier: The solution to the labeling problem is found in data simulations using the Monte Carlo method. There exists a profound knowledge of the particle interaction in the atmosphere: Given the energy and direction of some parent particle (gamma, proton, etc.) its interaction can be described by a probabilistic model

which gives a probability for particle collisions, possibly resulting in secondary particles. Each of these secondary elements may further interact with other particles of the atmosphere. This results in a cascade of levels of interactions that form the air shower. Figure 4 shows the transition of a particle and its interaction in the atmosphere. The showers previously shown in Figure 1 are examples of such simulated cascades. Unfortunately, the simulation of non-gamma showers is far more computationally extensive. Charged particles do interact with the atmosphere much more intense, resulting in more complex cascades. The simulation of atmospheric showers is performed in ray-tracing like software systems, most popular being the CORSIKA simulator [17]. The output is a simulated air shower, which needs to be run through a simulation of the telescope and camera device to produce the same raw input data as if the shower has been recorded using the real telescope.

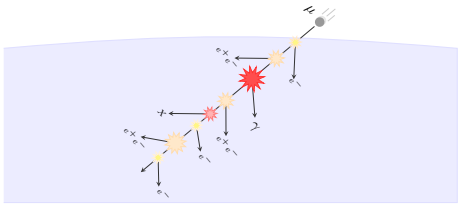

Fig. 4: Synthetic data by simulation in a stochastic process. Collision probabilities and generation of secondary particles are based on domain knowledge.

2.4 The Interdisciplinary Gap in Process Development

Looking at the big picture of the data analysis in a telescope like FACT, there is a steady development of each of the steps in progress: new features are tested for improved separation, different classifiers are investigated. The complete process from data recording to final energy estimation is continuously improved under the aspect of physics, typically resulting in diverse proprietary software solutions. The machine learning and computer science community, on the other hand, has produced a valuable collection of open-source software libraries for learning (e.g. MOA [7], WEKA [16] or RapidMiner [20]) and stream-lined process execution (e.g. Apache Storm, Samza). Unfortunately, the integration of these tools often requires specifically trained developers to adapt them to an application domain, which hinders the rapid prototyping evolution of the analytical domain software.

We generally refer to this problem as the *interdisciplinary gap* – the difficulty to apply sophisticated tools in a specific cross-disciplinary application domain. Over the collaborative research center project C3, we focused on bridging this gap by building a process design framework that provides the high-level means to define analysis chains from an end-user point of view, while keeping the power to integrate state-of-the-art software platforms such as the ones mentioned above.

3 Online Data Analysis for the FACT Telescope

After we provided a big picture of the analysis chain for the FACT telescope, we now present the *FACT-Tools*, our high-level approach to implementing an analysis chain from data acquisition to deriving the final results. The implementation focuses on an online processing of the recorded events, geared towards matching the data rate of the telescope. We build upon the streams framework, developed at the TU Dortmund University, which features a declarative XML specification of generic data flows. By providing a small set of application specific components that match the streams programming API, the domain experts gain full control of the overall process layout while retaining the possibility to integrate state-of-the-art machine learnign libraries and a possible mapping of the resulting data flows to large scale execution engines, like Apache Storm.

3.1 The streams Data Flow Framework

Modern data processing chains – like the one required for the FACT telescope – can generally be presented by their data flow. In such data flow graphs, a source emitting a sequence of records is linked to a graph of nodes, each of which provides the processing of input and the delivery of some output. Such data flow graphs are inherent to all modern data processing engines, especially in the field of data stream processing.

With the streams framework we aim at finding an appropriate level of abstraction that allows for the design of data flows independent of a specific execution engine. The focus of streams is a light-weight middle-layer API in combination with a descriptive XML-based specification language, that can directly be executed with the streams runtime or be mapped to topologies of the Apache Storm engine. The predominant focus in the development of streams was a simplistic light-weight API and process definition that is easily applicable by domain experts and adaptable to a variety of different use cases.

Each of the connected processes in streams consists of a pipeline of user-functions, which are applied to the processed items in order. Figure 6 shows a brick-like visualization of a process as pipeline of functions. The source nodes provides a sequence of data items that is individually processed by the process pipeline of user-functions. These user-functions are typically implemented in

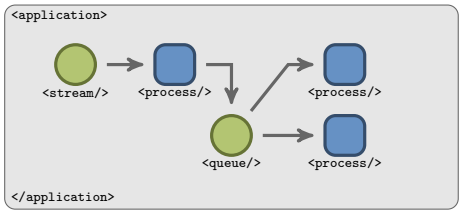

Fig. 5: The outline of an application in streams – a graph of connected processes.

single Java classes and are directly referenced by their implementation name from inside the XML element of the corresponding process. The figure shows functions for the calibration, image cleaning and extraction of Hillas parameters. With these features available, a classifier model can be applied.

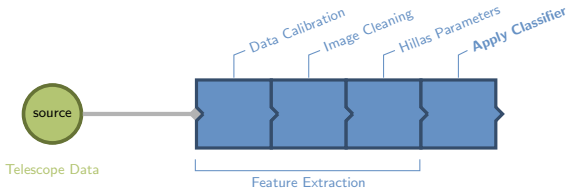

Fig. 6: A data stream *source* connected to a process that consists of four different user functions for calibration and feature extraction.

To achive the maximum level of flexibility, the data items (or tuples) that are passed from one user-function to the next, are wrapped in a simple hashmap (or dictionary), that can be accessed and enriched by each user-function. The data that can be stored in each of these can be of arbitrary serializable types, allowing for the implementation of user-functions that work on simple data types as well as frames/images (video processing) or telescope event data as we will discuss in the next section. To further ease the modelling of data flows with a set of implemented user-functions, the streams approach facilitates an automatic mapping of XML attributes to classes following the JavaBean conventions. This removes any intermediate layer between process modelling and function implementation.

```
<process input="telescope:data">
  <fact.data.DrsCalibration calibrationFile="file:/data/calib.fits" />
  <fact.image.ImageCleaning energyThreshold="2.45" />
  <fact.image.features.HillasParameters />
  <streams.weka.Apply modelUrl="http://sfb876.de/rforest.weka" />
</process>
```

Fig. 7: The XML corresponding to the pipeline of the previous figure.

3.2 FACT-Tools: Processing Telescope Data

The FACT-Tools is a collection of input implementations and user-functions that is built around the processing of telescope data. By implementing the required functionality in the context of the user-functions API of streams, this allows physicists to easily create data flows by XML specifications and test their processes in a reproducible manner. This rapid prototyping and library-style coding

led to a quickly evolving setting that covers the complete data analysis chain from data acquisition to evaluation.

The primal data gathered by the telescope is encoded in the FITS file format. The *Flexible Image Transport System* (FITS) is a file format proposed by NASA to store satelite images and other information in a compact, yet flexible way as it supports a variety of basic data types that can be stored. The bulk of data, recorded by the telescope for each event, is provided as a large array of values, sampled from the camera pixels. Along with those samples, additional information on the event, such as the number of the recording run, the time and high-resolution arrival times for each pixel. The FACT-Tools provides a `fact.io.FitsStream` implementation that reads this data from any input stream and emits a sequence of items (one per shower event). Table 1 shows an excerpt of the elements provided for each item.

Name (key)	Type	Description
`EventNum`	`Integer`	The event number in the stream
`TriggerNum`	`Integer`	The trigger number in the stream
`TriggerType`	`Integer`	The trigger type that caused recording of the event
`Errors`	`Integer`	Indicates communication errors
`UnixTimeUTC`	`Integer`	Timestamp of the recorded event in millisecond accuracy
`Data`	`Double[432000]`	The raw data array ($1440 \cdot 300 = 432000$ float values)

Table 1: The representation of a shower event as hashmap in the streams model.

User-Functions for Telescope Events

The core element of each event is found as key `Data` in Table 1. It holds the values sampled from each pixel upon a trigger. As a region of interest of 300 slices is written to disk for each event, this amounts to 432000 values. Based on calibration data, the sampled values need to be adjusted to match the specific voltage offsets and gains for each pixel, which may vary depending on the temperature and other environmental factors. Based on the calibrated per-pixel time series, a couple of additional user-functions can be applied, each of which reflects the computation of features such as the identification of shower- and non-shower pixels, the fitting of an ellipse to the supposed shower image and the derivation of geometric Hillas parameters or other properties of the event.

Geared towards the rapid prototyping of this data-preprocessing flow, a wide range of additional user-functions has been implemented by the domain experts, which range from additional time-calibrations to data corrections such as the removal of broken pixel data. Each of these preprocessing step is focused on improving the data quality and finding of features that may further improve the overall gamma-hadron separation task.

3.3 Integration of Machine Learning Libraries

As part of the abstract data flow design, we integrated the MOA and WEKA machine learning libraries as modules into the streams framework. Though especially MOA is geared towards online learning, the setting of the telescope data demands more for an *online application* of models: The data that is used for training the models, is synthetically generated and available as a batch data set. Typically, the size of that data is also comparably small, once the features for training have been extracted (the majority of the data volume is embodied in the raw data). A crucial aspect for the application of machine learning models, is the fact that only features, which are extracted *online* are suitable for use in such models. Any features that relate to an overall property of a data set, e.g. a normalization with respect to the sum computed over a set of instances, will not match the stream-lined setting that we aim for in the FACT data analysis.

The two user-functions `streams.weka.Train` and `streams.weka.Apply` have been implemented, which can be used to incorporate the training and application of WEKA models directly within a streams data flow. This ensures, that the same preprocessing setup can be used to feed the training of the model as well as its later application. The `Train` function collects a batch of user-specified instances to build its training data set. This is crucial as some features such as nominal type features require additional meta-data to be equal during training and model application. Upon building the classifier, the `Train` function outputs the serialized model in addition to the meta-data information about the attributes. In addition, its `features` parameter allows for an easy wild-card selection of features that shall be used for building the classifier. Any keys prefixed with an `@` character (e.g. `@id` or `@source`) are not regarded regular features for training the classifier. Figure 8 shows the XML setting of a process for training a random forest classifier using WEKA within streams. Classifier options are automatically mapped to XML attributes using Java's reflection API. The approach directly supports any of the provided WEKA classifiers. The corresponding `Apply` function has previously been shown in Figure 7. It requires a `modelUrl` parameter that holds the location of the serialized model. The streams framework automatically handles different URL types, such as file, http or classpath resources. This eases the sharing of processes and their models as well as a distributed execution of multiple instances of the analysis with a shared separation model.

```
<process input="simulator:data">
  <!--  preprocessing left out due to out space limitations  -->
  <streams.weka.Train
             features="*,!hillas:size"
           classifier="weka.classifiers.tree.RandomForest"
             numTrees="100"
               output="/data/random-forest.weka" />
</process>
```

Fig. 8: Training a WEKA classifier specified in XML.

4 Experiments

We tested the use of WEKA within the overall processing chain of the FACT telescope as modelled with the FACT-Tools. The data we used in the experiments was generated by Monte Carlo simulations using the CORSIKA software. The dataset contains 139333 shower events, of which 100000 events stem from gamma and 39333 events from proton (hadronic) particles. The events are simulated in raw data format and passed through the standard cleaning and feature extraction chain using the FACT-Tools, resulting in 83 features suitable for separation. For the experiments, we focused on the following aspects:

1. Predictive performance of the classifier for gamma-hadron separation,
2. Improvements of separation by re-organisation of the data flow,
3. Throughput performance of the overall processing chain.

An interesting note for the performance comparisons is the optimization criterion used to assess the classification. Whereas the traditional machine learning community often uses precision, recall or accuracy for grading classifier performances, the physics field is more interested in a pure sample of gamma ray induced events. A well-accepted measure in this area is the *Q-factor* defined as

$$Q = \frac{\varepsilon_\gamma}{\sqrt{\varepsilon_p}} \quad \text{with} \quad \varepsilon_\gamma = \frac{N_\gamma^{det}}{N_\gamma} \quad \text{and} \quad \varepsilon_p = \frac{N_p^{det}}{N_p}$$

where ε_γ and ε_p represent the *gamma efficiency* (number of gammas detected divided by total number of gammas in dataset) and the *proton* or *hadron efficiency* respectively. The Q-factor aims at assessing the purity of the resulting gamma events. In addition we also provide the significance index [15].

4.1 Gamma-Hadron Separation with WEKA Classifiers

Using the basic Hillas parameters and an additional set of features build up on these, we tested different classifiers: Random Forests, an SVM implementation and a Bayesian filter. Table 2 shows the classification performance for these approaches. Each classifier was evaluated with a 10-fold cross validation and optimized parameters: The Random Forest was trained with 300 trees with 12 features and a maximum depth of 25. The SVM used an RBF kernel with $k_\gamma = 0.014$ and $C = 10$. The training set in each fold was balanced.

Classifier	Q Factor	Significance	Accuracy	Precision
Random Forest	4.796 ± 0.178	65.55 ± 0.358	0.969 ± 0.0021	0.959 ± 0.0029
SVM	4.013 ± 0.916	60.227 ± 1.859	0.953 ± 0.010	0.936 ± 0.025
Naive Bayes	2.267 ± 0.0609	51.65 ± 0.503	0.841 ± 0.0048	0.864 ± 0.0062

Table 2: Performances for gamma/hadron separation with different classifiers.

For an improved purity, the classification is weighted with the confidence provided by the classifier. Those *confidence cuts* are applied by physicists to obtain an even cleaner sample as is crucial for all subsequent analysis steps. All *gamma* predicted elements with a confidence less than some threshold are regarded as proton predictions. Though this increases the number of missed gamma events, it eliminates false positives, which may tamper with subsequent steps such as energy estimation. Figure 9 shows the impact of these confidence cuts.

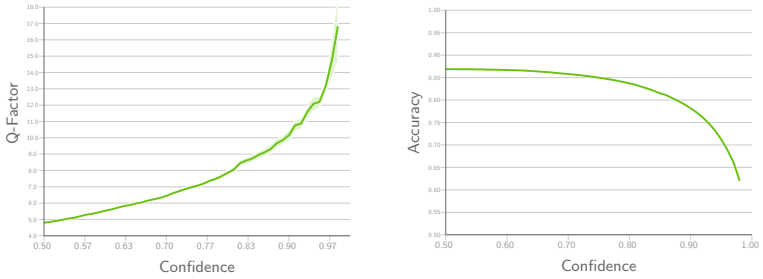

Fig. 9: Refined selection by *confidence cuts*, which improves the purity at the cost of missed signals, reflected in a decreased recall, which diminishes the accuracy.

4.2 Signal Separation with Local Models

A parameter describing the "intensity" of the shower is `size`, which incorporates the area of the ellipse and the voltage levels of the covered pixels. The `size` parameter is known to highly correlate with the energy of the original particle [6] and allows for a grouping of events based on their energies. We investigated the separation performance of Random Forests, trained on disjoint datasets defined on a partitioning using the $\log_{10}$(`size`) feature (Figure 10).

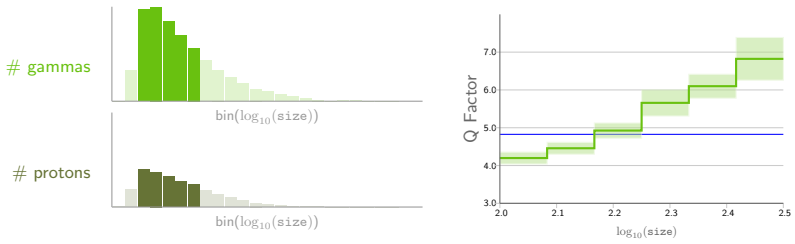

Fig. 10: Distribution of gamma and hadronic events over `size` range (left) and performance of local models per bin vs. global model (right).

We limited this experiment to bins of $\log_{10}(\texttt{size})$, which had at least 10.000 events for testing. The right plot shows the Q-Factor for models trained and evaluated on separate bins (green) and the global model trained over data from all bins (blue) without any confidence cuts applied. The figure provides the Q-Factor averaged over a 10-fold cross validation, the light green area shows the standard deviation. We only looked at the Q-Factor here, as it is the accepted criterion in that community.

4.3 Throughput Performance of the FACT-Tools

The FACT telescope records at a rate of 60 events per second, where each event amounts up to 3 MB of raw data, resulting in a rate of about 180 MB/s. Figure 11 shows the average processing time (milliseconds) of the user-functions for the complete analysis in a *log-scale*. The first two blocks of functions reflect the bulk of raw data processing. Ellipse fitting and other feature extractions which are input to the classification step are shown in bright green. The process is able to handle the full data rate of the telescope on a small scale Mac Mini.

The improved separation by the use of local models suggests a split of the data stream. Though the `size` feature is only available at a later stage in the process, it highly correlates with properties available directly after the data calibration (2nd user function) has been applied. In combination with the local models this allows for a massive parallelization by data stream grouping, when deploying the process in distributed environments such as Apache Storm. The generic abstraction provided by `streams` already allows for a direct mapping of the XML process specification to a Storm topology.

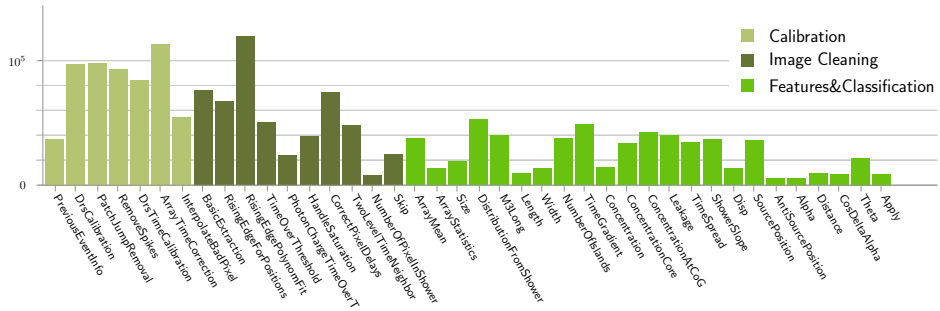

Fig. 11: Average processing times of the analysis chain functions (log scale).

5 Summary and Conclusion

In this work we reviewed the analysis chain of an Air-Cherenkov telescope and provided an online implementation of that process by the high-level abstraction

framework streams. The resulting data flow covers the complete data processing and feature extraction. It integrates previously trained WEKA models and is able to handle the data rate of the FACT telescope in real-time. By focusing on a declarative, easy-to-use abstraction, we lowered the barrier of end-users designing their own analytical data flows, enabling direct use of state-of-the-art machine learning libraries. The generic nature of the streams abstraction and its embeddability allows for a direct integration into large scale distributed streaming engines and sets the scene to cope with the load of upcoming, more high-resolution telescopes.

Future work will further focus on improving the separation power and investigating models for energy estimation for FACT. As the scalability aspect obviously touches the data preprocessing we are looking into a direct mapping of user-functions defined using the streams API in Apache Storm and Apache Hadoop, aiming at a full code re-use without modifications.

We are also confident that this use-case can be mapped to other scenarios as we successfully tested it in steel-mill factories [24] and smart cities [3,25].

Acknowledgement This work has been supported by the DFG, Collaborative Research Center SFB 876, project C3 (`http://sfb876.tu-dortmund.de/`).

References

1. A. U. Abeysekara et al. On the sensitivity of the HAWC observatory to gamma-ray bursts. *Astroparticle Physics*, 35:641–650, May 2012.
2. H. Anderhub et al. Fact - the first cherenkov telescope using a g-apd camera for tev gamma-ray astronomy. *Nuclear Instruments and Methods in Physics Research A*, 639:58–61, May 2011.
3. Alexander Artikis, Matthias Weidlich, Francois Schnitzler, et al. Heterogeneous stream processing and crowdsourcing for urban traffic management. In *Proceedings of the 17th International Conference on Extending Database Technology*, 2014.
4. R. Atkins et al. Milagrito, a tev air-shower array. *Nuclear Instruments and Methods in Physics Research*, 449:478–499, 2000.
5. H.M. Badran and T.C. Weekes. Improvement of gamma-hadron discrimination at tev energies using a new parameter, image surface brightness. *Astroparticle Physics*, 7(4):307 – 314, 1997.
6. K. Berger, T. Bretz, D. Dorner, D. Hoehne, and B. Riegel. A robust way of estimating the energy of a gamma ray shower detected by the magic telescope. *Proceedings of the 29th International Cosmic Ray Conference*, pages 100–104, 2005.
7. Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, August 2010.
8. R.K. Bock, A. Chilingarian, et al. Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2–3):511 – 528, 2004.
9. Christian Bockermann and Hendrik Blom. The streams framework. Technical Report 5, TU Dortmund University, 12 2012.
10. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

11. S. Bussino and S.M. Mari. Gamma–hadron discrimination in extensive air showers using a neural network. *Astroparticle Physics*, 15(1):65 – 77, 2001.
12. Bradley W. Carroll and Dale A. Ostlie. *An Introduction to Modern Astrophysics*. Addison-Wesley, San Francisco: Pearson, 2nd (international) edition, 2007.
13. Mathieu De Naurois. Analysis methods for atmospheric cerenkov telescopes. *arXiv preprint astro-ph/0607247*, 2006.
14. E. Faleiro, L. Muñoz, A. Relaño, and J. Retamosa. Discriminant analysis based on spectral statistics applied to TeV cosmic γ/proton separation. *Astroparticle Physics*, 35:785–791, July 2012.
15. S. Gillessen and H. L. Harney. Significance in gamma-ray astronomy - the li and ma problem in bayesian statistics. *Astronomy and Astrophysics*, 430(1):355–362, November 2004.
16. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.
17. D. Heck, J. Knapp, J. N. Capdevielle, G. Schatz, and T. Thouw. *CORSIKA: a Monte Carlo code to simulate extensive air showers.* Forschungszentrum Karlsruhe GmbH, Karlsruhe (Germany), February 1998.
18. D. B. Kieda and VERITAS Collab. Status of the VERITAS ground based GeV/TeV Gamma-Ray Observatory. In *High Energy Astrophysics Division*, volume 36 of *Bulletin of the American Astronomical Society*, page 910, August 2004.
19. A. M. Hillas. Cerenkov light images of EAS produced by primary gamma rays and by nuclei. In F. C. Jones, editor, *Proceedings of the 19th International Cosmic Ray Conference*, volume 3, pages 445–448, La Jolla, August 1985.
20. Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.
21. D. Petry et al. The MAGIC Telescope - prospects for GRB research. *Astronomy & Astrophysics Supplement Series*, 138:601–602, September 1999.
22. G. Pivato et al. Fermi LAT and WMAP Observations of the Supernova Remnant HB 21. *The Astrophysical Journal*, 779:179, December 2013.
23. B.M. Schäfer, W. Hofmann, H. Lampeitl, and M. Hemberger. Particle identification by multifractal parameters in γ-astronomy with the hegra-cherenkov-telescopes. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 465(2–3):394 – 403, 2001.
24. Jochen Schlüter, Hans-Jürgen Odenthal, Norbert Uebber, Hendrik Blom, Tobias Beckers, and Katharina Morik. Reliable bof endpoint prediction by novel data-driven modeling. In *AISTech Conference Proceedings*. AISTech, 2014.
25. Francois Schnitzler, Alexander Artikis, Matthias Weidlich, et al. Heterogeneous stream processing and crowdsourcing for traffic monitoring: Highlights. In *Proceedings of the European Conference on Machine Learning (ECML), Nectar Track*, pages 520–523. Springer Berlin Heidelberg, 2014.