# User documentation

## Common Setup prerequisites

In order to run all of the four primary applications, user has two options.

1. Download the archived version of the solution accessible on the link:[]. If you decide to use this option, no further instalation is needed for any of the programs in the whole solution.
2. Clone the the solution:

* The operating system supports *.NET Framework* version *4.7.2*. For further information about *.Net Framework,* visit <https://docs.microsoft.com/en-us/dotnet/framework/get-started/system-requirements>
* The library ClusterCalculator is downloaded (cloned). This library is a part of the solution Cluster Processor available at [https://gitlab.mff.cuni.cz/teaching/nprg045/mraz/Celko2020](https://gitlab.mff.cuni.cz/teaching/nprg045/mraz/Celko2020%20)

## Cluster Viewer

***Setup and input***

**Requirements:**

* Classifier for Clusters project, which is part of the repository at [https://gitlab.mff.cuni.cz/teaching/nprg045/mraz/Celko2020](https://gitlab.mff.cuni.cz/teaching/nprg045/mraz/Celko2020%20)
* Third-party software for three-dimensional plotting Chart Director for .Net library accessible at [https://www.advsofteng.com/download.html](https://www.advsofteng.com/download.html%20)

**Application Dependencies:**

Cluster Viewer

Cluster Calculator

Classifier

Trainer

ClassifierForClusters

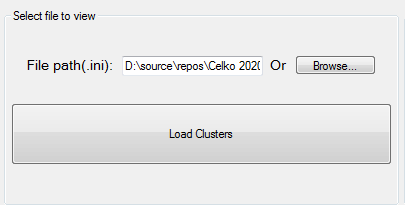
Descr Generator

Cluster Filter

ClassifierUI

Cluster Experiment

To view clusters, we run the viewer and either type the path to our .ini file or click the Browse button, select the desired .ini file and click the Load Clusters button.



2. Ini file which will be loaded

3. Load selected clusters

1.Select the ini file

If the message “File was loaded successfully”appears, it means that our collection of clusters is now ready for viewing. If we do not see this message, it means that a problem occurred during the loading of the file. The error is further specified by the given message. Some of the common causes are:

* .ini file does not exist or is inaccessible
* .cl and .px files referenced by the .ini file do not exist or are inaccessible
* .cl, .px or .ini file is not in the correct format

For more information about the input file format, see the section [].

The image displays the energy of each pixel logarithmically mapped to the color spectrum starting from white (for pixels with energy below 2eV) through yellow (for pixels with energy greater than 2eV and less than 15eV) to orange and red (pixels with energy up to 500eV). If the same pixel is hit twice in the cluster, the pixel with more energy is displayed.



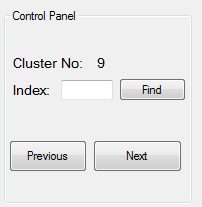
Pixel Energy around 3keV

Pixel Energy around 20keV

Energy more than 200keV

***Browsing***

After the cluster collection is loaded, we can navigate through the collection using the Previousand Nextbuttons. To find the -th cluster in the collection, we can select the index of the cluster in the Control Panel box and click the Findbutton.



***Histograms***

Apart from collection browsing, users can also view histograms. There are two histograms available in the viewer:

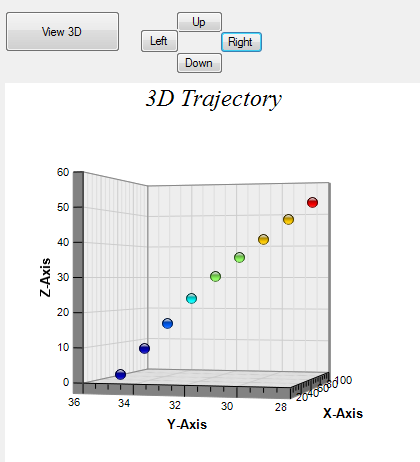
* Collection Histogram displays the histogram of the whole collection for the pixel count attribute.



* The Pixel Histogram represents the histogram of the pixels of the currently viewed cluster for the energy.



Both histograms are calculated and displayed after clicking Show Collection Histogram (or Show Pixel Histogram).

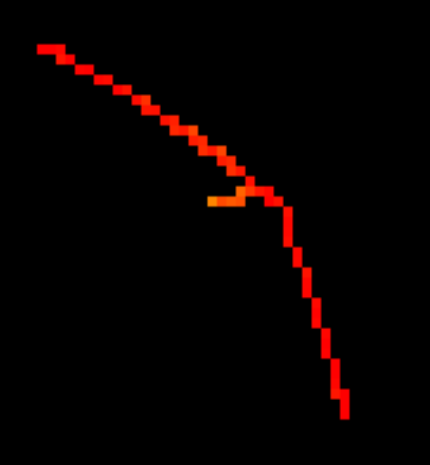
***3D visualization***

To get a better idea of how the trajectory of the cluster looked like, we can visualize the cluster in 3D. The visualization is based on the energy of each pixel. To view the three-dimensional image of the cluster user can click the View 3D button. After the image is displayed, the user can rotate the image around the x-axis and z-axis by clicking the Up, Down, Left, and Right buttons.

***Cluster Attributes***

We use the button Show Attributes located in the Cluster Details section to calculate cluster properties. By clicking Show Attributes, we can view various attributes of the cluster in the JSON file format. These attributes range from straightforward like Total Energy and Maximum Energy to the more sophisticated ones like Branch Count and Relative Halo Size. For further information about the cluster attributes, please see the section [insert section].

***Skeletonization***

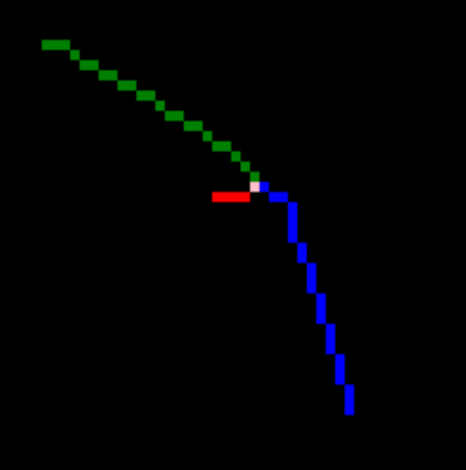
 Skeletonization often referred to as a thinning process, “transforms an input binary image into a skeleton by reducing the original image which contains different thicknesses to a thin representation (a set of curves and lines).“ Skeletonization can be used as a tool for removing the halo effect of the clusters while also preserving the shape of the original cluster. During the skeletonization process, as the image is thinning, pixel energy is split among its neighbours. This means the total energy of the cluster and its skeletonized version is the same. To view the skeleton of the original cluster, we click Skeletonize.

Skeletonization

***Branch analysis***

After we can find the skeleton of the cluster, we can try to detect particle trajectories in the cluster. To do so, we can click the Show Branches button. The center point of the cluster is represented by the white dot, while the separate branches are denoted by the distinct colors – blue, red, and green. Each branch can have its sub-branches - the starting point of the sub-branch is contained in its parent branch. The sub-branches are highlighted by the lighter shade of their parent branch color.

Branch 1



Branch 2

Branch 3

Center

***Classification***

Using all the features, we are able to calculate about the cluster, we can classify the cluster into various categories. This classification is done via machine learning using neural networks. The default classes of particles that are implemented in the classifier are:

* Lead, Iron, and Helium
* muons, electrons, pions, and protons.

In many cases, the classification process is complex, so we cannot expect the class prediction to have 100% accuracy. To find out more about the classification process and its reliability, see the section [insert section]. In the viewer, user can load a classifier by clicking Browse and choosing the right trained classifier in .csf file.

## Filter

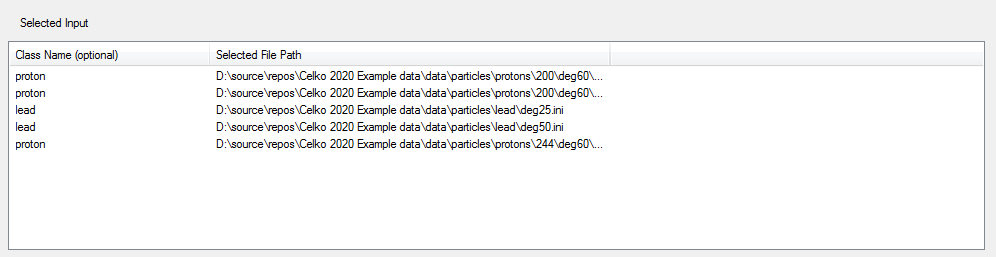
The user interface of the Cluster Filter application is very similar to the Cluster Viewer. As the first step, we select the input .ini file. This can be done either by typing out the path to the file or clicking the Browse button. Then, we choose the name for the output .ini file. After that, we can select the properties to filter by and set the lower and upper bounds for the property. If no bound (upper or lower) is specified, the filter automatically sets the bound to the maximum (or minimum) possible value. To start the processing, we click the Process button. During the filtering process, a new .cl file is created. The newly created file contains only those clusters, which fit the filtering criteria. However, no .px file is created because the output reuses the original pixel file to speed up the calculation and prevent unnecessary copying. After the filtering is done, the filter displays the message “Filtering successfully completed”. If any other message is shown, it means there was an error. The message should provide more information about the problem. In most cases, it is either the incorrect data format or file inaccessibility.

## Description Generator

The Cluster Description Generator is a tool for the attribute calculation for the whole collection of clusters at once. This can be useful when creating training data for the machine learning algorithms.

1. We start by clicking the Browse and Add… button, where we select one or more .ini files and add them to the Selected Input collection.
2. To add more input files, we can repeat the process until all desired files are shown in the Selected Input box.
3. To remove elements in the collection, we click the Remove Selected button.
4. For every single file we select, we can edit its Class Name column by triple-clicking – there can be multiple files containing the same class. This way, we set the name of the particle present in the input file, which can then be used for the supervised machine learning algorithms.

“Proton“ class partitions



“Lead“ class partitions

1. Then we can choose the output file name in the Select output text field and also tick the attributes that will be calculated. To include the cluster class name as an attribute, remember to tick the Class attribute.
2. After that, we can choose either the **even** distribution of each class in the output or the distribution **proportional** to the particular class size.
3. For each file with given class (further referenced to as a class partition), we can choose whether we want to process those in **a serial order** (provided by the order on input – next partition is processed after the previous was already fully processed) or in **parallel order** (after a single cluster from current partition is processed, a cluster from the next partition is set to be processed next).
4. The user can also select the **ending condition** of the process. When this condition is satisfied, the program finishes the calculation – for a large number of clusters, the process may take several minutes to complete. There are three types of ending condition:
   1. Any partition is fully processed (**First partition**)
   2. Any class is fully processed (**First class**)
   3. All classes are fully processed (**Last class**)

By default, no cluster on the input will appear in the output more than once. When generating the data for imbalanced classes (there are huge differences in class sizes), this could lead to machine learning problems with the classification of the less frequent classes. To compensate for that, we can choose the Align class. By selecting the align class, any other class partition that is processed will not be discarded but will be processed again until the specified Align class is fully processed.

## Classifier

The ClassifierForClusters is a console application that provides an interface for the classification process of the selected clusters. The prerequisite for the classifier is to have Accord.net NuGet packages installed. [10]

Syntax:

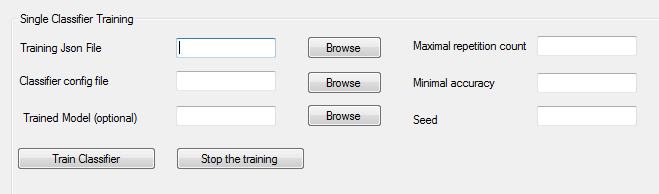
**ClassifierForClusters.exe [trained\_classifier.csf] [file\_to\_classify.json] [options]**

|  |  |
| --- | --- |
| Command line options | |
| **--simple** or **--multi** | Specification of the classifier type (single-layered or multi-layered), default is single |
| **--distr** | Print only key-value pairs of the class name and its frequency. |
| **--classes** | Split the clusters into separate JSON files according to the predicted class and also print key-value pairs of the frequencies. In order to view the results in viewer later, cluster must contain attributes ClFile, PxFile and ClIndex. |
| **--specials** | Split the clusters into separate JSON files (--classes), print frequencies (--distr) while also create a file for non-trivial unclassified clusters. |

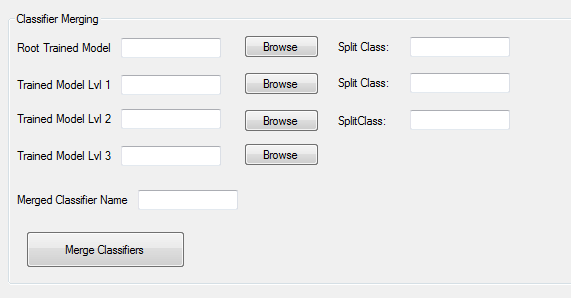
## ClassifierUI

Classifier UI is a tool which enables training classifier models based on the input parameters. User needs to set:

* Training Json file = training data in JSON format
* Classifier config file = parameters of the classifier in JSON format []
* Trained model = If we want to continue in training existing model, we fill in this field with .csf file.
* Minimal Accuracy = When this level of accuracy (or better) is reached on validation set, the classifier is stored into a file. I tis a value between 0 and 1.
* Maximal repetition count = The maximal number of times we will try to train our classifier to reach minimal accuracy
* Seed = Integer value, based on the seed data is split randomly into training and validation set.



The UI also provides an option to merge up to 4 simple trained classifiers into a multi-level classifier. User can choose the root classifier which will be applied first, then choose a class (split class) which will be fed to the Trained Model Lv1 for further classification. That mean for combining classifiers we need to fill in split classes because bottom level classifier cannot have a split class. If you use only 2 classifiers (root and Lvl 1) make sure all the other fields for classifiers are empty.



On the example below we can see multi-level classifier which uses three simple classifiers and two split classes.

## ClassifierTrainer

ClassifierTrainer is a console application which can train a classifier, same as ClassifierUI, only has a different API.

Syntax:

**ClassifierTrainer.exe [training\_data.json] [network\_parameters\_config.json] [options followed by their value (space separated)]**

Example

ClassifierTrainer.exe lead\_training data.json lead\_network\_config\_parameters.json --acc 0.8 --seed 42 --maxrep 5

|  |  |
| --- | --- |
| Command line options | |
| **--trained** | (Optional) Indicates existing .csf file should be trained again instead of creating a new classifier. A path to the trained model should follow this option. |
| **--acc** | Sets the minimal accuracy of the classifier on the validation set to save the result. Must be followed by the accuracy value between 0 and 1. |
| **--seed** | Sets the seed, according to this seed the data split into training and validation is done. It is followed by an integer. |
| **--maxrep** | Sets the max number of learning attempts on the dataset until the minimal accuracy (or better is achieved) |