

AMUSE 0.4b beta (Advanced MUSic Explorer) User Manual

Igor Vatolkin

October 29, 2025

RWTH Aachen University
Department of Computer Science
Chair for AI Methodology (CS 14)
Theaterstraße 35-39
52062 Aachen
Germany

`igor.vatolkin[AT]rwth-aachen.de`
`https://www.aim.rwth-aachen.de`
`http://sig-ma.de`

Contents

1	Introduction	3
1.1	License Notes	3
1.2	Repository and Requirements	3
1.3	Remarks on History	3
1.4	Citing AMUSE	4
1.5	Acknowledgements	4
1.5.1	Developers	4
1.5.2	Financial Support	4
2	Backgrounds	5
2.1	Machine Learning Pipeline in AMUSE	5
2.1.1	Feature Extraction	5
2.1.2	Feature Processing	6
2.1.3	Model Training	7
2.1.4	Model Application	8
2.1.5	Validation	8
2.1.6	Optimization	9
2.2	The Attribute-Relation File Format	9
3	Installation	10
3.1	Preliminary Notes for MacOS	10
3.2	In the Shell	10
3.3	In Eclipse	10
3.4	First Steps	11
4	Graphical User Interface	13
4.1	AMUSE Wizard	13
4.2	Preferences	13
4.2.1	General Settings	13
4.2.2	Logging	14
4.2.3	Grid Settings	15
4.2.4	Manage Plugins	15
4.2.5	Annotation Settings	16
4.3	Experiment Configurator	16
4.3.1	Feature Extraction Configurator	17
4.3.2	Feature Processing Configurator	19
4.3.3	Model Training Configurator	19
4.3.4	Model Application Configurator	21
4.3.5	Validation Configurator	21
4.3.6	Optimization Configurator	23
4.4	Annotation Editor	23
4.4.1	Single Track Annotation Editor	24
4.4.2	Multiple Tracks Annotation Editor	26

1 Introduction

1.1 License Notes

AMUSE (Advanced MUSic Explorer) is an open-source Java framework for various music data analysis and music information retrieval tasks. It is developed within the Chair for AI Methodology, RWTH Aachen University ¹, headed by Prof. Dr. Holger Hoos, together with the Computational Intelligence research group at the Chair of Algorithm Engineering, Department of Computer Science, TU Dortmund University², headed by Prof. Dr. Günter Rudolph.

AMUSE is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. AMUSE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License³ along with AMUSE.

1.2 Repository and Requirements

AMUSE is available as the GitHub repository⁴. The framework itself is in the folder:

- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amuse>

The AMUSE plugins which extend the functionality of the basis master branch are in the following folders:

- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amusePluginChromaToolbox>
- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amusePluginKeras>
- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amusePluginLibrosa>
- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amusePluginMIRToolbox>
- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amusePluginSonicAnnotator>
- <https://github.com/AdvancedMUSicExplorer/AMUSE/tree/master/amusePluginHarmonicFeatures>

This manual is CURRENTLY UNDER DEVELOPMENT and does not contain the complete information how to use AMUSE. We plan to release an updated manual soon.

You should have a Java 8 OpenJDK or a newer version and JavaFX installed on your system. AMUSE should also run with Oracle Java 1.8 or newer on most systems.

Some of the AMUSE components and plugins use Matlab and Python. However, they are not required to run AMUSE in general.

The current version was tested on Ubuntu Unix, MacOS, and Windows systems.

1.3 Remarks on History

The following versions of AMUSE were previously released:

- **0.1:** The first version presented at ISMIR [16].
- **0.2:** Integration of the annotation editor for individual tracks (e.g., marking time events or segments) and multiple tracks (e.g., assigning genre or emotion tags).
- **0.3:** Support of fuzzy / multi-class / multi-label classification, plugins to Librosa [9] and Keras [1], as well as many further improvements (the version presented at SIGIR [15]).
- **0.4:** The current version with easier configuration using AMUSE workspace and more classification methods.

¹<https://www.aim.rwth-aachen.de/>

²<https://ls11-www.cs.tu-dortmund.de/rudolph/start>

³<http://www.gnu.org/licenses>

⁴<https://github.com/AdvancedMUSicExplorer/AMUSE>

The GitHub commits distinguish between different code contributions:

- **BUGFIX**: A bug fix / correction of wrong behavior
- **UPDATE**: An update to an existing functionality, such as refactoring or code optimizations
- **NEW**: A new feature
- **EPIC**: A substantial update such as the first implementation of the annotation editor

1.4 Citing AMUSE

If you use AMUSE for your research, please cite one of the following publications:

- **THE FIRST PRESENTATION OF AMUSE AT ISMIR 2010**: I. Vatulkin, W. Theimer, and M. Botteck: AMUSE (Advanced MUSic Explorer) – A Multitool Framework for Music Data Analysis. Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR), pp. 33-38, 2010 [16].
- **A RECENT OVERVIEW OF THE MOST RELEVANT UPDATES TO AMUSE AFTER 2010**: I. Vatulkin, P. Ginsel, and G. Rudolph: Advancements in the Music Information Retrieval Framework AMUSE over the Last Decade. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 2383-2389, 2021 [15].

1.5 Acknowledgements

1.5.1 Developers

Many contributions to the source code and substantial improvements were done by student and research assistants at the TU Dortmund University: Polina Kozarovytska (since 2024), Clara Pingel (2021-2024), Philipp Ginsel (2018-2022), Frederik Heerde (2017-2018), Fabian Ostermann (since 2015), Daniel Stoller (2011-2015), and Clemens Wältken (2008-2011).

1.5.2 Financial Support

The AMUSE development was partly supported within the following projects:

- **MUSICDESCRIBER: PERCEPTIONAL MUSIC CONTENT ANALYSIS**
Funded by Nokia Research Center Bochum
2006–2008
- **MULTI-OBJECTIVE OPTIMIZATION OF AUTOMATIC MUSIC CLASSIFICATION BASED ON HIGH-LEVEL FEATURES AND COMPUTATIONAL INTELLIGENCE METHODS**
Funded by Klaus Tschira Foundation
2009–2013
- **EVOLUTIONARY OPTIMIZATION FOR INTERPRETABLE MUSIC SEGMENTATION AND MUSIC CATEGORIZATION BASED ON DISCRETIZED SEMANTIC METAFEATURES**
Funded by German Research Foundation (DFG)
2018–2021

2 Backgrounds

2.1 Machine Learning Pipeline in AMUSE

AMUSE allows you to perform all algorithmic steps from the general machine learning pipeline, namely the extraction of features, their processing, training of classification models, their application and validation, as well as optimization of algorithm parameters. Each step is independently run by a corresponding AMUSE node and is called *task* in the following. The basic AMUSE tasks are:

- **FEATURE EXTRACTION:** Extraction of audio features (e.g., Mel frequency cepstral coefficients [13] using `javaudio` library [8]).
- **FEATURE PROCESSING:** Application of further methods to prepare the feature inputs for model training (e.g. normalization of feature values to a given range, application of the principal component analysis [6], selection of time frames based on musical events).
- **MODEL TRAINING:** Building of a classification model (e.g., Random Forest using WEKA library [2]) for music tracks with already processed features and annotated labels.
- **MODEL APPLICATION:** Application of the previously created model to classify new music tracks.
- **VALIDATION:** Estimation of classification measures (e.g., confusion matrix) to assess the performance of the previously created model.
- **OPTIMIZATION:** Search for optimal algorithm parameters (e.g., length of classification frames which are assigned to genres).

This pipeline is visualized in Figure 1. AMUSE stores the result of each step in the corresponding “database” which is a folder in AMUSE workspace, except for predictions of classification models.

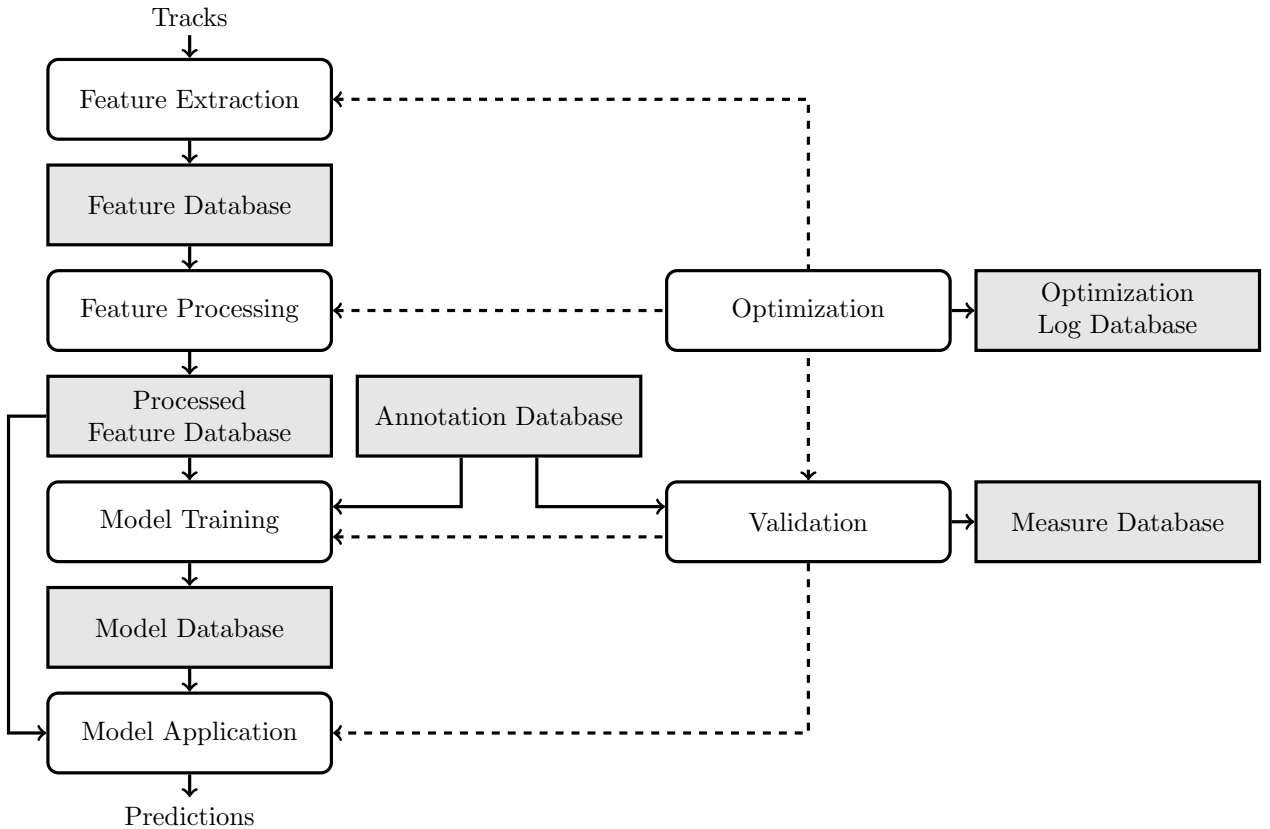


Figure 1: The machine learning pipeline in AMUSE.

2.1.1 Feature Extraction

Feature extraction stores numeric properties of audio signals which can be used later for music classification and data analysis. These properties are extracted from different domains, such as time domain, spectrum, or cepstrum. Some of them belong to rather low-level signal descriptors (like the number of zero-crossings), others

are more interpretable (pitch class profiles or beats).

It is distinguished between the following kinds of features:

- **WINDOWEDNUMERIC**: A feature with a numeric value which is extracted from the frames of the same length and with the same step size, e.g., zero-crossings extracted from frames of 512 samples with no overlap (step size is equal to 512 samples). Note that features which are extracted from the complete audio track (like its duration in seconds) also belong to this category, with the frame length set to -1 samples.
- **WINDOWEDSTRING**: A feature with a string value which is extracted from the frames of the same length and with the same step size, e.g., the predominant chord.
- **EVENT**: Individual time events: onsets, beats, or segment boundaries.
- **SEGMENTEDNUMERIC**: A feature with a numeric value which is extracted from extraction frames of a variable length, like the share of vocals in a music segment.
- **SEGMENTEDSTRING**: A feature with a string value which is extracted from extraction frames of a variable length, like the chord progression.

Each feature has a unique AMUSE ID. However, it is possible to define additional configurations, e.g., for different extraction frames. After the extraction, the features are stored in the AMUSE feature database as ARFFs, for instance, as:

$$\underbrace{\text{/home/user/AmuseWorkspace/Features/ACDC/Back.in.Black/Hells.Bells/}}_{\text{Path to feature database}} \underbrace{\text{Hells.Bells.}}_{\text{Relative path to music track}} \underbrace{\text{400}}_{\text{Track name}} \underbrace{\text{.arff}}_{\text{ID}}$$

Note that it is not recommended to use white spaces in file names, as some of feature extractors may fail in that case. You may consider to apply `sanity.pl`⁵ script before feature extraction with AMUSE.

Some features require the previous installation of the corresponding plugin.

2.1.2 Feature Processing

The goal of feature processing is to construct classification windows from previously extracted features. As these features may be extracted from different frames, in the first step the harmonized feature matrix is estimated [14], as sketched in Figure 2.

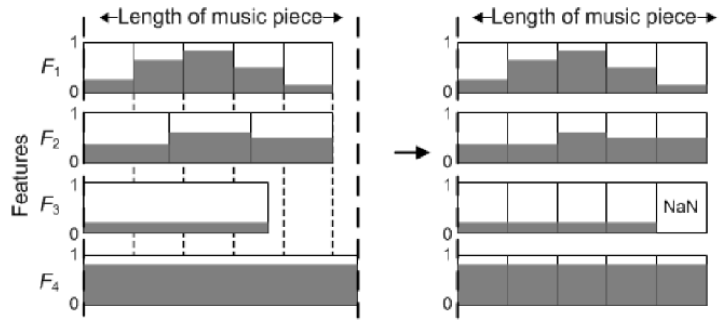


Figure 2: An example of the harmonization of feature matrix [14, p. 373].

Here, the shortest extraction frame length across all features is calculated (feature F_1), and all other feature values are also mapped to these frames. For features extracted from longer frames, the same values may contribute to several consecutive shortest frames, as for features F_2 – F_4 . In case the boundary of a longer frame is exactly in the shortest frame, the value from the longer frame with the largest overlap to the shortest extraction frame is stored. When no overlap between the longer frame and smallest frame exists (as for the last shortest frame and feature F_3), the “Not a Number” (NaN) value is stored.

Now, various processing methods can be applied which operate on feature or time dimension of the harmonized feature matrix:

⁵<https://github.com/splitbrain/sanity/blob/master/sanity.pl>

- **PREPROCESSING** methods like normalization or replacement of missing values prepare the features for the application of further processing methods and typically do not change the dimensionality of the harmonized feature matrix.
- **FEATURE DIMENSION PROCESSING** methods typically reduce the number of features, for instance, selecting a limited number of principal components [6] or applying a feature selection strategy removing irrelevant and redundant features [4]. In some cases, feature dimension processing may increase the dimensionality, if new features are constructed by the application of mathematical operators for the original feature dimensions as applied for music data in [12, 11].
- **TIME DIMENSION PROCESSING** methods usually select some frames with regard to musical events (e.g., storing only frames with or between beat events or only frames from the middles of music segments like intro, verse, and chorus). Further, more enhanced concepts like structural complexity [7, 3] can be applied.
- **AGGREGATION OF CLASSIFICATION WINDOWS** is a final step which estimates final feature vectors for each classification window. This can be done using a simple model like the mean and standard deviation for each feature dimension within the classification frame, or also applying more complex time series models which store, e.g., the coefficients of multiple linear regressions over the original feature time series [10].

2.1.3 Model Training

The model training task is responsible for the building of models which assign music data to categories or predict some numerical properties from others. In general, we may distinguish between following method types:

- **SUPERVISED CLASSIFICATION** requires not only processed features, but also annotations (labels) provided by experts or users. Then, the classifier creates some rules which predict these annotations from new unlabeled data.
- **UNSUPERVISED CLASSIFICATION** assigns data to different groups or clusters without provided annotations. This kind of methods is currently not available in AMUSE but will be integrated in near future.
- **REGRESSION** does not assign categorical labels to processed features directly but estimates a numerical property (a feature or also a numeric label) from one or more other numerical properties. Regression is planned to be integrated in AMUSE in future.

For the ground truth annotations and predictions, it is distinguished between two relationship types:

- **BINARY, OR CRISP RELATIONSHIP** describes a processed feature vector as completely belonging or not belonging to a category, i.e. the relationship grade $y \in \{0, 1\}$.
- **CONTINUOUS, OR FUZZY RELATIONSHIP** describes to which extent a processed feature vector belongs to a category, i.e. the relationship grade $y \in [0, 1]$. As an example, a “progressive rock oper” track may have two relationships $y_{CLASSICAL} = 0.4$ and $y_{ROCK} = 0.8$ (note that as these values do not describe probabilities, they must not produce 1.0 in their sum).

For classification tasks, there exist following strategies to predict the labels:

- **SINGLE-LABEL** predictions assign a sole label to processed features in a binary way, i.e., rating the feature vector as “positive” (belonging to a category or group) or “negative” (do not belonging to a category a group).
- **MULTI-CLASS** predictions assign a sole label to processed features selecting this label from several ones (e.g., assigning a music track to a genre classical, pop, or rock).
- **MULTI-LABEL** predictions assign one or more labels to processed features.

Table 1 provides an overview of the classification algorithms and their support of model types.

Table 1: Support of model types by classification algorithms.

Algorithm	Single-label	Multi-class	Multi-label	Binary	Continuous
k -nearest neighbors	Yes	Yes	No	Yes	No
Gradient boosted trees	Yes	Yes	No	Yes	No
Fuzzy k -nearest neighbors	Yes	Yes	Yes	Yes	Yes
J48	Yes	No	No	Yes	No
Naive Bayes	Yes	Yes	No	Yes	No
Random forest	Yes	Yes	No	Yes	No
Support vector machines	Yes	No	No	Yes	No

2.1.4 Model Application

After the training, models can be used to classify new tracks. Again, the same feature extraction and feature processing steps that have been used prior to classification training need to be applied to the tracks that should be classified. The model receives these data and predicts labels.

It can be distinguished between window-level and track-level predictions: window-level predictions are assigned to classification frames (typically of several seconds for musical genre recognition), and track-level predictions to complete music tracks. In the latter case, the decision is done by majority voting (the label with the larger number of related classification frames is assigned to the complete track).

Figure 3 shows examples for window-level predictions using the model previously trained to detect Pop/Rock genre. We may observe, for instance, that almost all Rap tracks can be correctly classified by majority voting, but they still contain quite a large number of classification windows which were wrongly assigned to Pop/Rock.

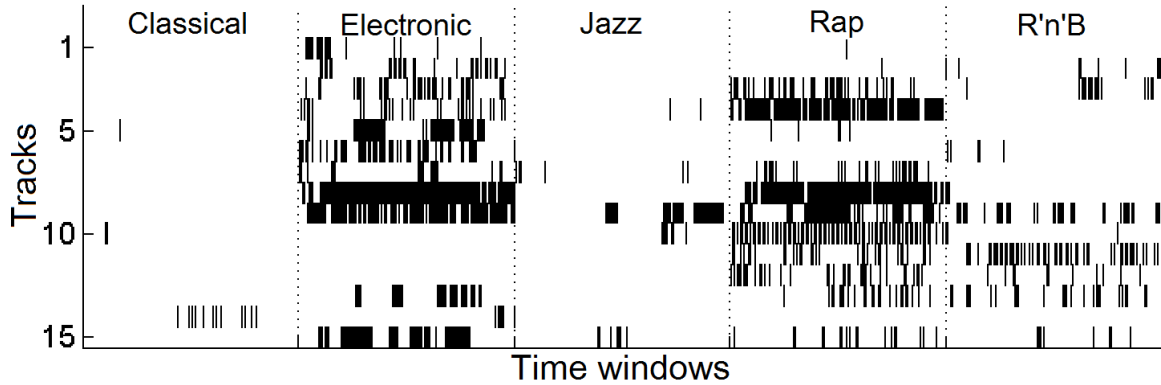


Figure 3: An example for window-level predictions (black dashes, correspond to 4s classification windows; the width is normalized w.r.t. track lengths) predicting Pop/Rock from [17, p. 346].

2.1.5 Validation

The validation task estimates one or more values (usually numerical) for the evaluation and comparison of trained models. For the measurement of quality, an annotated dataset is needed to check the correctness of predictions. In that case, it must not be used to train the model, to avoid overfitting against this particular dataset.

It is distinguished between the following types of evaluation measures:

- **DATA REDUCTION**: Measures which describe the impact of data reduction after feature processing, compared to the original feature matrix.
- **CONFUSION MATRIX**: Numbers of true positives, true negatives, false positives, and false negatives.
- **CORRELATION-BASED MEASURES**: Measures based on the correlation between annotated and predicted labels.
- **MEASURED ERROR**: Numerical differences between annotated and predicted labels: mean squared error, absolute error, etc.

- **STANDARD MEASURES**: Common measures based on the confusion matrix: accuracy, precision, recall, etc.
- **TRACK LIST**: List of correctly predicted classification frames (not a measure).
- **EVENTS**: Measures for the evaluation of event prediction algorithms such as beats or onsets.

2.1.6 Optimization

In the whole machine learning pipeline, you will often find parameters that alter the behavior of each step. Optimization searches for the best configuration by executing these steps with different parameters and validating the resulting model.

2.2 The Attribute-Relation File Format

Most of input/output data in AMUSE are stored using the Attribute-Relation File Format (ARFF), which was developed for WEKA [2], a machine learning library that is also contained in AMUSE. It describes a table in which every column contains values of a certain data type. Figure 4 serves as an example for the ARFF.

```
@RELATION musicfiles

@ATTRIBUTE Id NUMERIC
@ATTRIBUTE Genre STRING
@ATTRIBUTE Type {instrumental, 'vocal music'}
@ATTRIBUTE 'Duration in Minutes' NUMERIC

@DATA
0, 'Pop', 'vocal music', 3.5
1, 'Classic', 'instrumental', 8.1
9, 'Hip-Hop', 'vocal music', 4.0
% This is a comment and will be ignored.
12, 'Rock', 'instrumental', 2.2
```

Figure 4: An example for an ARFF file. Each color indicates the affiliation of an attribute and its values.

An ARFF file is divided into two parts.

The first part of the file starts with **@RELATION** tag, followed with a freely selected name of the table. Then, the header is defined: each column is described by a line starting with **@ATTRIBUTE**, followed with the attribute name and its data type. There are three different data types commonly used in AMUSE: numeric, string, and nominal.

- **NUMERIC** denotes attributes with numerical values (decimals or whole numbers).
- **STRING** expresses the eponymous data type and is used for values consisting of any number of characters.
- **NOMINAL** defines a limited set of possible value types in curly brackets, only these values are allowed for this attribute.

Whenever a name contains a blank space, it must be put in quotation marks, like “Duration in Minutes”. Otherwise, quotation marks are optional.

After every column was defined, the second part of the file starts, indicated by the line **@DATA**. From now on, each line represents a row in the table, with a value for every previously defined column (attribute), likewise ordered and separated by a comma.

Comments start with **%** and are ignored when reading the ARFF file. For further documentation, visit https://waikato.github.io/weka-wiki/formats_and_processing/arff_stable/.

3 Installation

3.1 Preliminary Notes for MacOS

On MacOS, some steps may be required to set up the permissions for a proper execution of AMUSE:

- **STARTING FROM THE TERMINAL:** A disc access permission from “Settings → Privacy and Security Settings → Full Disc Access” should be given to the terminal application.
- **USING ECLIPSE:** Permissions are required to see audio files etc. If you are installing Eclipse for the first time, a pop-up comes up asking for permission to reach the files on the desktop. And if it is still not working or your files are somewhere else, running the following command can help with giving the necessary security permissions: `sudo codesign --force --deep --sign -/Applications/Eclipse.app`.

3.2 In the Shell

You can get the current version from the GitHub repository using the command:

```
git clone https://github.com/AdvancedMUSICExplorer/AMUSE.git
```

3.3 In Eclipse

First, please check that the compiler compliance level is set to 1.8:

1. Click on “Window” → “Preferences”
2. Go to “Java” → “Compiler”
3. “Compiler compliance level” must be set to 1.8

To checkout the project with git, please follow these steps:

1. Click on “File” → “Import...”
2. Select “Git” → “Projects from Git”
3. Select “Clone URI”
4. Enter the repository location `https://github.com/AdvancedMUSICExplorer/AMUSE.git`.
5. A dialog as shown in Figure 5 should appear. Mark the branch(es) you want to load. Mark only “master”, if you want to get only the code intended for users.

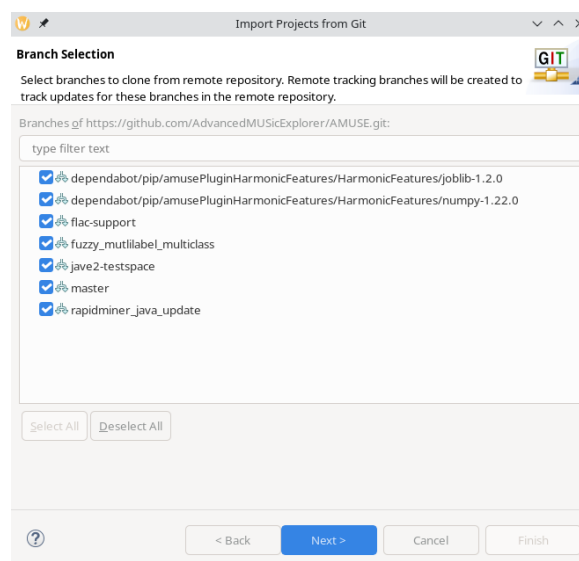


Figure 5: The dialog to select the branches that should be cloned.

6. In the next dialog, choose where AMUSE should be saved as seen in Figure 6.

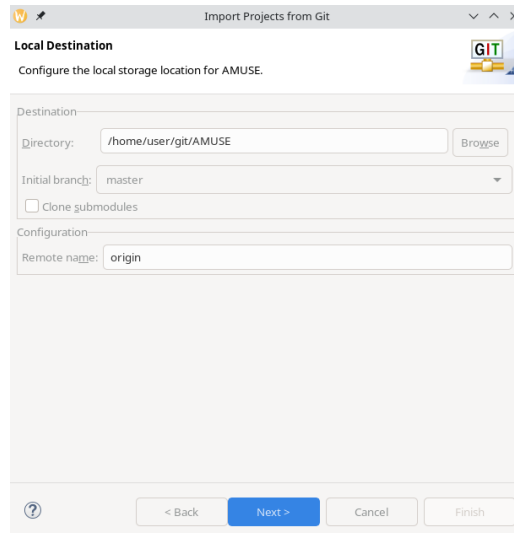


Figure 6: The dialog to select the folder that should be checked out.

7. Select “Import existing Eclipse projects”
8. In the next dialog (as seen in Figure 7), select the the project(s) you want to load. Mark only “amuse”, if you want to get the code from the main project only.

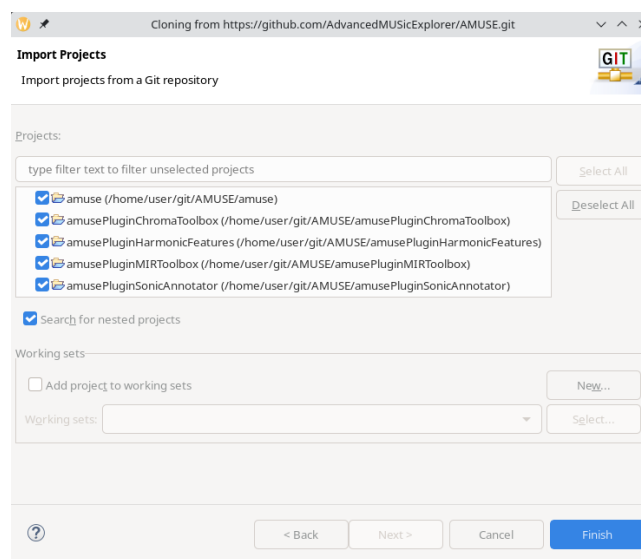


Figure 7: The dialog to select the folder that should be checked out.

9. Click on “Finish”.
10. Depending on the Java distribution used, it may be necessary to add the libraries included in JavaFX manually. To do so, right-click the AMUSE project, select “Build Path”, then “Configure Build Path”, and “Add External JARs” in “Libraries”-tab. Finally, add the JARs from JavaFX folder.

3.4 First Steps

You can either start AMUSE from `amuseGUI.bat`, `amuseGUI.sh`, or from your development environment. In the last case, you should use `AMUSE.SCHEDULER.GUI.CONTROLLER.WIZARDCONTROLLER` as a main class. To ensure that AMUSE runs without issues you should add the environment variable “AMUSEHOME” to your run configuration with the path to your AMUSE folder and add “`-javaagent:lib/jar-loader.jar`” as an VM argument.

If you want to run AMUSE via the scripts and cannot start java just using a “java” command from your command line, please adjust the path to it in the files `amuseGUI.bat` (for Windows) or `amuseGUI.sh` (for Unix and MacOS).

In case you will receive a message “CodeCache is full”, please configure the arguments of the virtual machine, e.g., using `-XX:ReservedCodeCacheSize=25600K`.

After the first start, you should see a dialog that will ask you to go to AMUSE Settings and set the path to AMUSE workspace, cf. Figure 8. The path to AMUSE workspace should lead to an empty folder that was created previously by the user. AMUSE will then create subfolders for extracted features, models, etc. in the AMUSE workspace and will store files in these folders accordingly. You can change this and further settings at any time by clicking on the “Edit amuse settings” button in the start screen shown in Figure 9. See Section 4.2.1 for more details on settings.

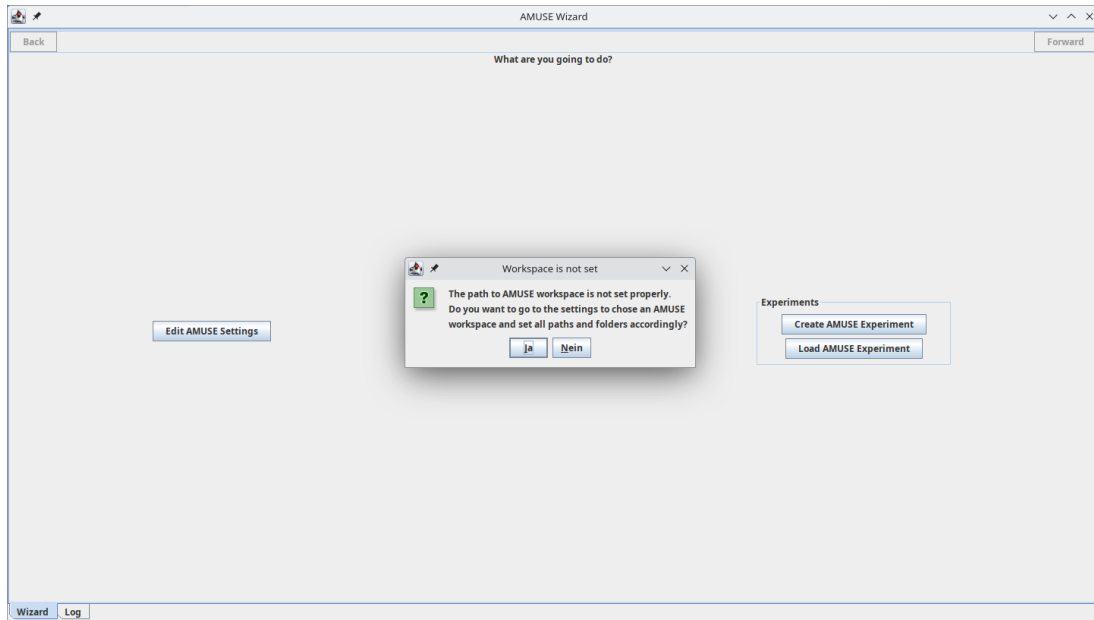


Figure 8: The dialog shown, when starting AMUSE for the first time.

4 Graphical User Interface

4.1 AMUSE Wizard

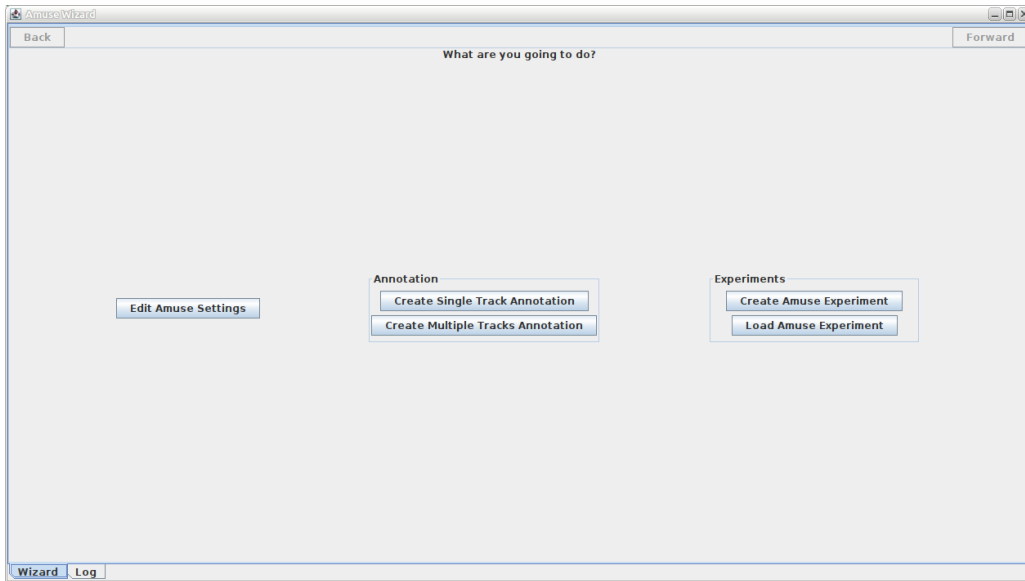


Figure 9: AMUSE start screen.

The start screen of AMUSE is shown in Figure 9. From here, you can access the different functionalities:

- **EDIT AMUSE SETTINGS:** Shows the screen for changing the preferences that is described in Section 4.2.
- **CREATE SINGLE TRACK ANNOTATION:** Opens the editor used for annotations of a single track (see Section 4.4.1).
- **CREATE MULTIPLE TRACK ANNOTATION:** Displays the editor used for annotations of multiple tracks (see Section 4.4.2).
- **CREATE AMUSE EXPERIMENTS:** Configure new experiments (see Section 4.3).
- **LOAD AMUSE EXPERIMENT:** Shows a dialog in which you can select an experiment configuration to load.

4.2 Preferences

4.2.1 General Settings

In Figure 10, the screen for the general settings is shown.

- **PATH SETTINGS:** Here, the path to AMUSE folder must be set as well to AMUSE workspace which contains several subfolders for input/output data (called AMUSE databases). In a default configuration of AMUSE workspace, all subfolders are set automatically. They can be individually configured using the flag “Show Advanced Path Settings”.
 - **MUSIC DATABASE** is a folder with music audio files—however it is also possible to work with the music files from other directories on your machine.
 - **FEATURE DATABASE** stores raw extracted features.
 - **PROCESSED FEATURE DATABASE** stores processed features.
 - **ANNOTATION DATABASE** is a folder for ground truth annotations.
 - **MODEL DATABASE** is a folder for classification models.
 - **MEASURE DATABASE** stores evaluation measures.
 - **OPTIMIZATION DATABASE** contains logs after experiments for the search of the optimal algorithm parameters.
- **EXTERNAL TOOLS:** Here, you can fill in the paths to your Java, Matlab, and Python executables.

- **WAVE-SAMPLING SETTINGS:** The default option is to downsample the given audio files to 22050 Hz and run stereo-to-mono conversion.
- **MISCELLANEOUS SETTINGS:** The log level defines the minimal level of importance that messages should have to be displayed in the log screen (see below). The maximal number of task threads can be changed if you want to make use of several processing units in your machine.

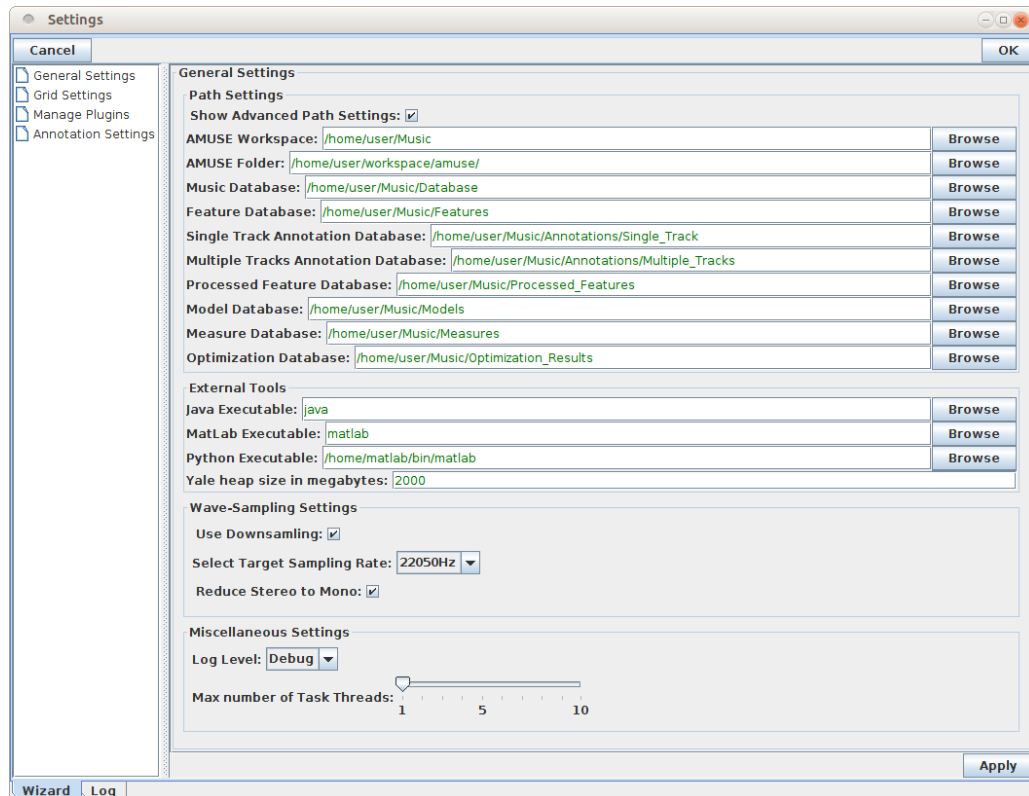


Figure 10: General settings.

4.2.2 Logging

In AMUSE, events like the completion of an experiments or errors that occurred will be written to the log. It is available both as text file in the AMUSE directory and in the GUI. When starting an experiment, you will be redirected to the logging screen automatically. You can also switch to it anytime by clicking on “Log” in the lower left corner.

There is a hierarchy of four log levels:

- **FATAL:** An error that hinders AMUSE to function properly any longer.
- **ERROR:** An error that prevented a computation from finishing successfully. It is often caused by an invalid experiment configuration. AMUSE can continue to work.
- **WARN:** The computation continues without errors, but some unexpected behavior might occur which the user is warned about.
- **INFO:** A piece of information like the completion of an experiment or its part.
- **DEBUG:** A detailed piece of information about internal steps.

In the preferences, you can select the minimum log level to be shown. Setting the log level to “Debug” can help to see more information if any problems occur. Fatal errors, errors, and warnings are always displayed.

An example for how the logging screen could look like is shown in Figure 11.

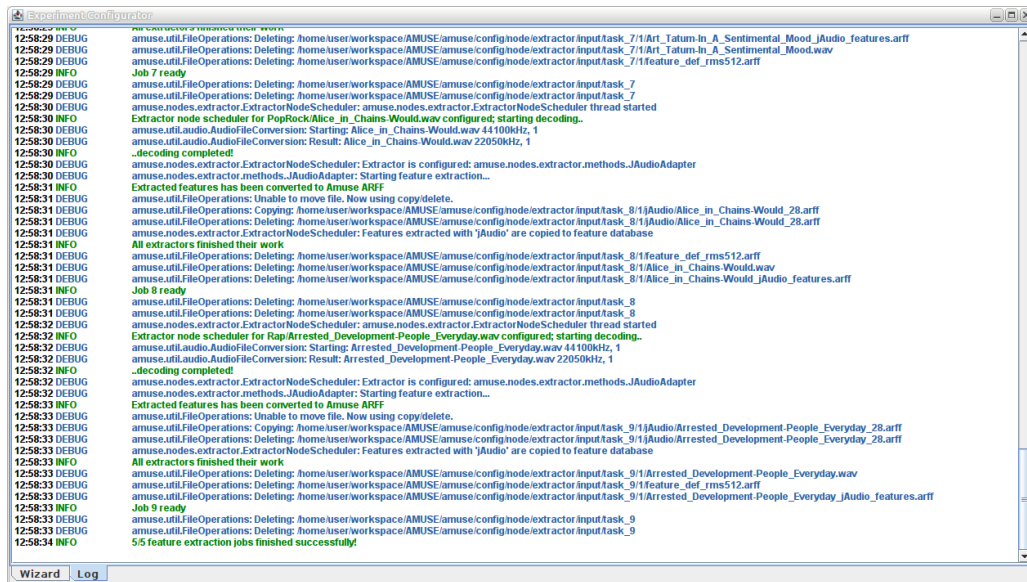


Figure 11: The logging screen.

4.2.3 Grid Settings

In case AMUSE should not directly start the experiments but forward them to a grid system like SLURM [5], you can enable this behavior using checkboxes in the grid settings, see Figure 12. Here, you can set up the paths to corresponding shell scripts.

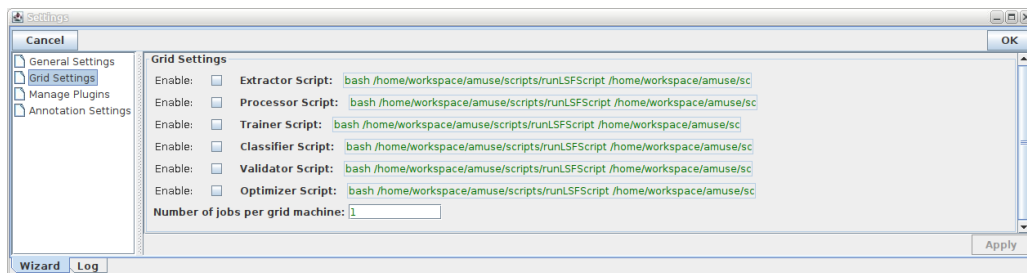


Figure 12: Grid settings.

4.2.4 Manage Plugins

As a framework, AMUSE can incorporate plugins to extend its functionality. They can be downloaded on GitHub just like AMUSE. Figure 13 shows the screen in which you can install or uninstall plugins. To install a new plugin, download the whole folder of the plugin from GitHub and click on “Install New”. Now, select the folder and confirm. Sometimes, you have to adjust the permissions in the plugin directory in AMUSE folder (e.g., setting UNIX binaries as “executable”), to omit errors occur during the usage of the plugins.

You can uninstall a plugin by selecting it and clicking on “Uninstall Selected”.

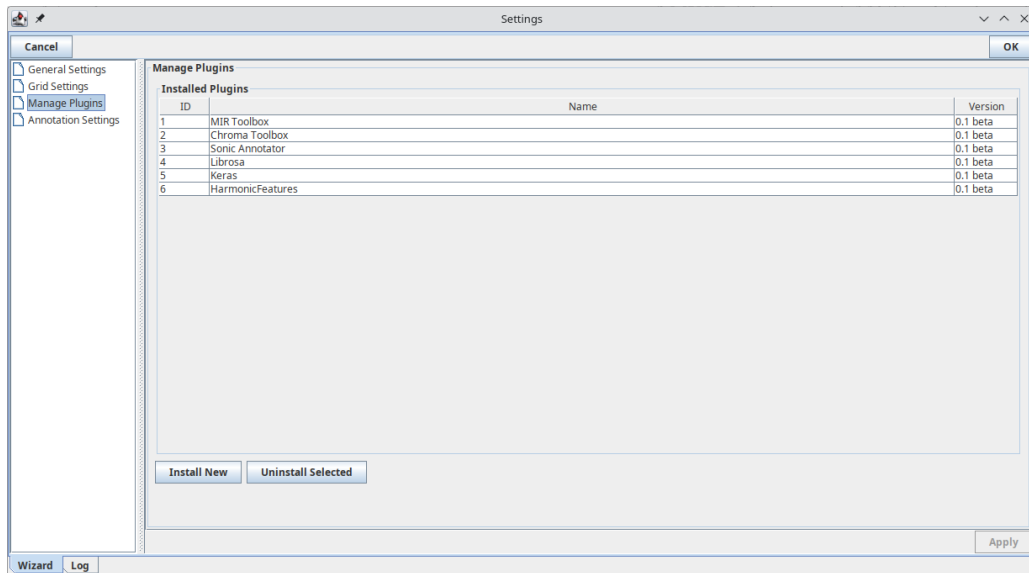


Figure 13: Manage plugins.

4.2.5 Annotation Settings

The settings for annotation editor are shown in Figure 14. You can select window- and hop sizes for the windows, for which the audio spectrum is extracted. Further, the narrow line on the audio spectrum that indicates the current position of the music (cf. Figure 25) can be hidden by unchecking the box “Mark Current Time In Audio Spectrum”.

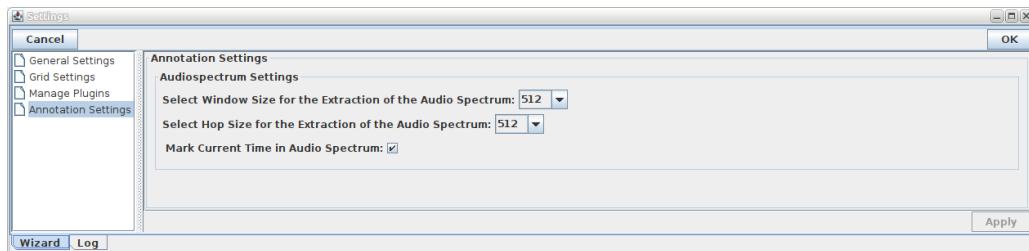


Figure 14: Annotation settings.

4.3 Experiment Configurator

With the Experiment Configurator, the single steps from the classification pipeline as described in Section 2.1 can be configured. The screen is shown in Figure 15.

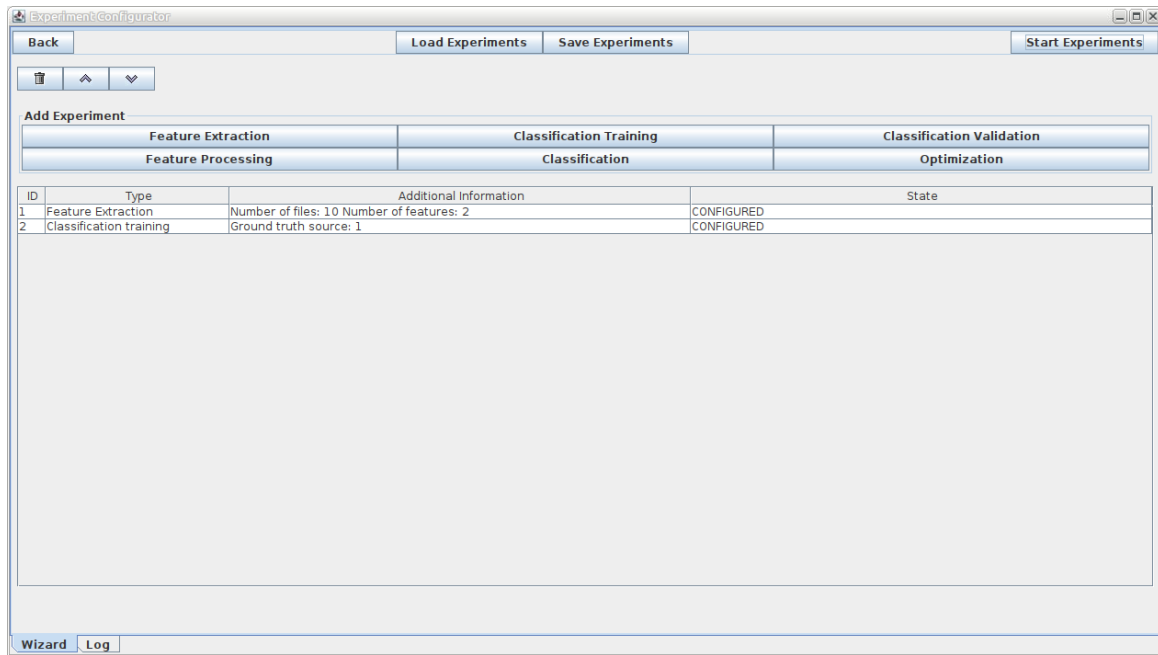


Figure 15: The Experiment Configurator.

In the Experiment Configurator, you can choose from the following actions:

- **BACK**: Go back to AMUSE start screen.
- **LOAD EXPERIMENTS**: Selects previously configured experiments.
- **SAVE EXPERIMENTS**: Saves the configured tasks in a user-defined file.
- **START EXPERIMENTS**: Starts the execution of the configured tasks in the defined order and switches to the logging screen that is described in Section 4.2.2.
- **ADD EXPERIMENT**: Adds a new task in table of the Experiment Configurator.
- : Removes the selected task.
- : Moves the selected task up.
- : Moves the selected task down.

In the column labeled “State”, the current state of the tasks is shown:

- **CONFIGURED**: The task is configured and ready to start.
- **RUNNING**: The task is currently running. No new tasks can be started while a task is already running.
- **IN_QUEUE**: The task is in the queue. Once the tasks in front of it are finished, it will start.
- **FINISHED**: The task is finished.

4.3.1 Feature Extraction Configurator

Screenshots for the Feature Extraction Configurator are provided in Figure 16 and Figure 17. On the left panel, you can select the music files. The right side displays all available features. This list is stored in the file “featureTable.arff” in the config folder from AMUSE and can be edited manually. For every specified audio file, features marked with a check will be extracted and saved in the feature database.

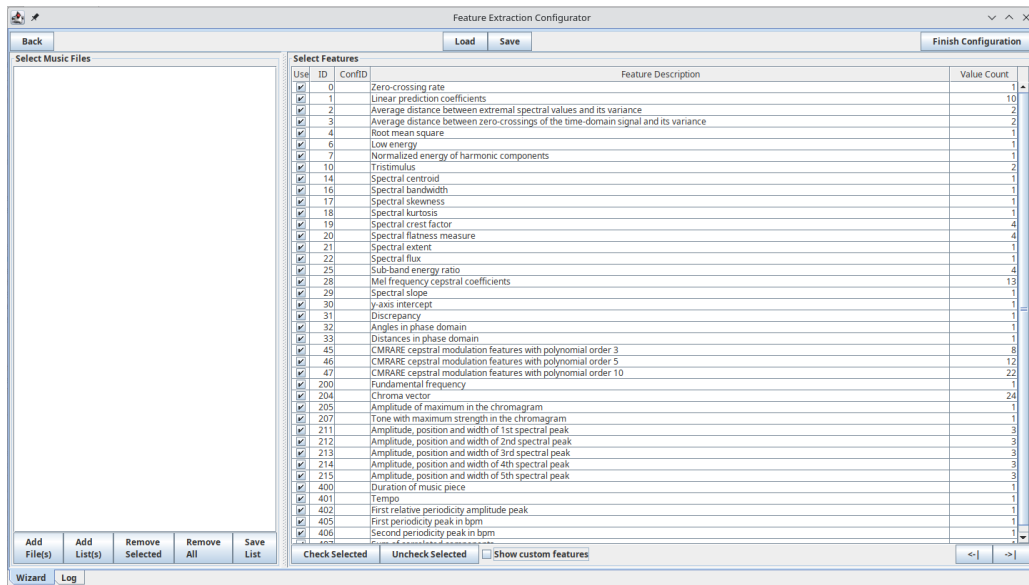


Figure 16: The Feature Extraction Configurator.

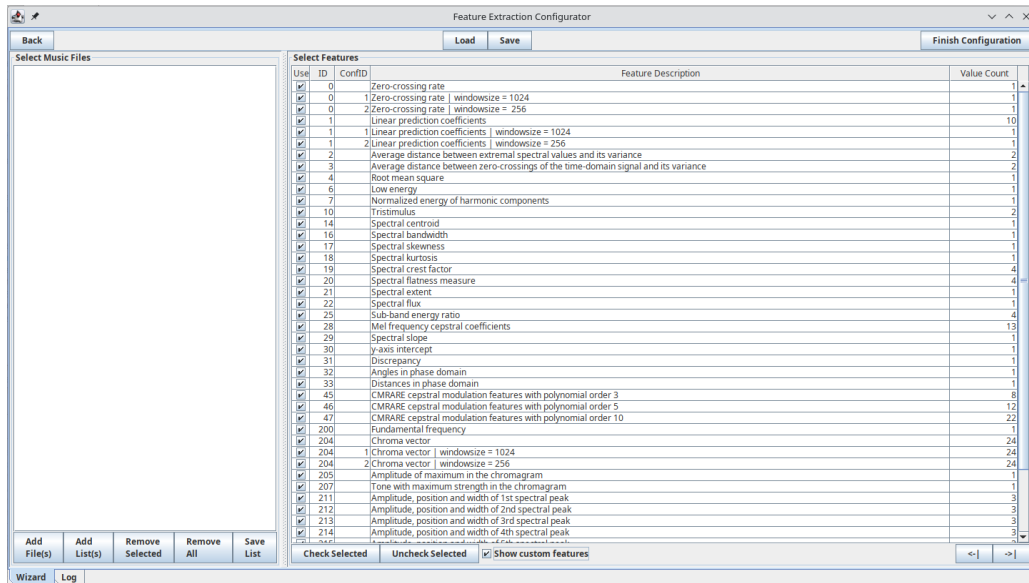




Figure 17: The Feature Extraction Configurator with custom features enabled.

The audio files can be selected with the following buttons:

- **ADD FILE(S)**: Add files using a file chooser dialog.
- **ADD LIST**: Add audio files from an ARFF file list.
- **REMOVE SELECTED**: Remove the selected files.
- **REMOVE ALL**: Remove all files.
- **SAVE LIST**: Save the current files as an ARFF file list.

The set of features for extraction can be loaded and saved by clicking on  and , respectively. Also, the whole feature extraction task can be loaded and saved with the “Load”- and “Save”-Buttons placed above the feature table. Checking the box “Show custom features” enables custom feature configurations. These custom feature configurations are defined in the folder “config/features” inside the AMUSE folder. The configurations are specified in ARFF files with the feature IDs as their names. Here, alternative extraction tools or extraction scripts can be set up for the features, modifying their extraction parameters, e.g. extraction window and step sizes or the number of dimensions.

4.3.2 Feature Processing Configurator

When processing features, AMUSE takes ones that have been extracted already and alters them by applying user-defined steps. You must select which features for which tracks should be processed. That is why the first view of the Feature Processing Configurator looks similar to the Feature Extraction Configurator in Figure 16. After finishing this selection, the processing steps can be defined in the view shown in Figure 18. Usually, the methods either operate on the feature dimension only (e.g., principal components analysis), or on the time dimension (selection of time windows with musical events like beats or onsets).

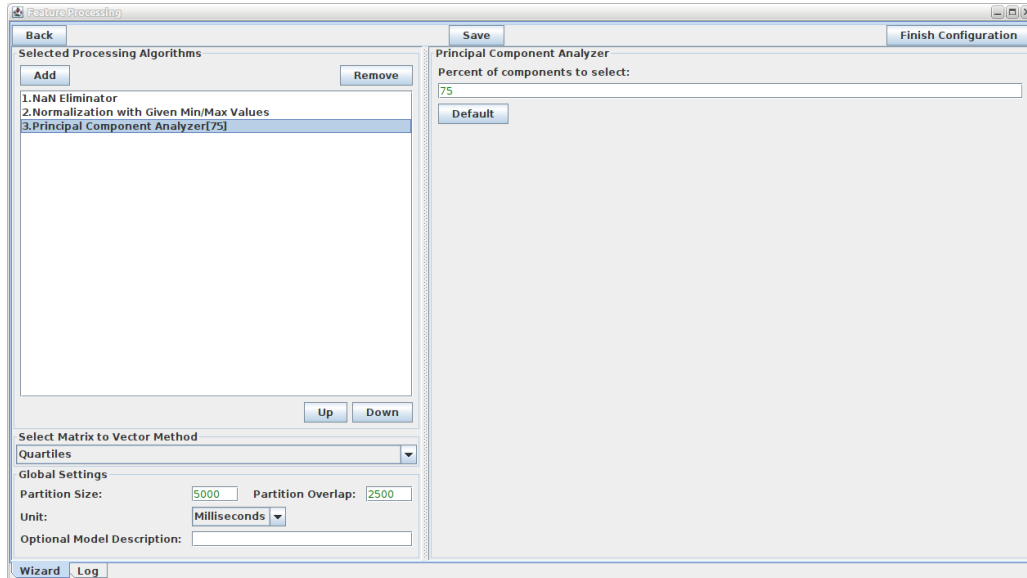


Figure 18: The Feature Processing Configurator.

In the upper left corner, you can add a step by clicking on “Add”. To change the order of processing methods, please select a step and use the “Up”- and “Down”-Buttons. On the right side, the individual parameters of the currently selected step can be configured. After applying these steps, the feature matrix is constructed as described in Section 2.1.2, using “Partition Size” and “Partition Overlap” parameters. A special matrix-to-vector conversion method is “RawFeatures” which does not harmonize the extraction time windows but only combines the feature vectors of these windows into large vectors. This method can be useful when you want to perform certain processing steps but do not want to lose data by unifying the windows.

You can find all available processing steps and their IDs in the file “processorAlgorithmTable.arff”. The conversion methods are listed in the file “processorConversionAlgorithmTable.arff”. Both files are located in the config folder from AMUSE.

4.3.3 Model Training Configurator

For model training, the left side of the Classification Training Configurator shown in Figure 19 is used to describe the data set and preprocessing, and the right side to setup the training method.

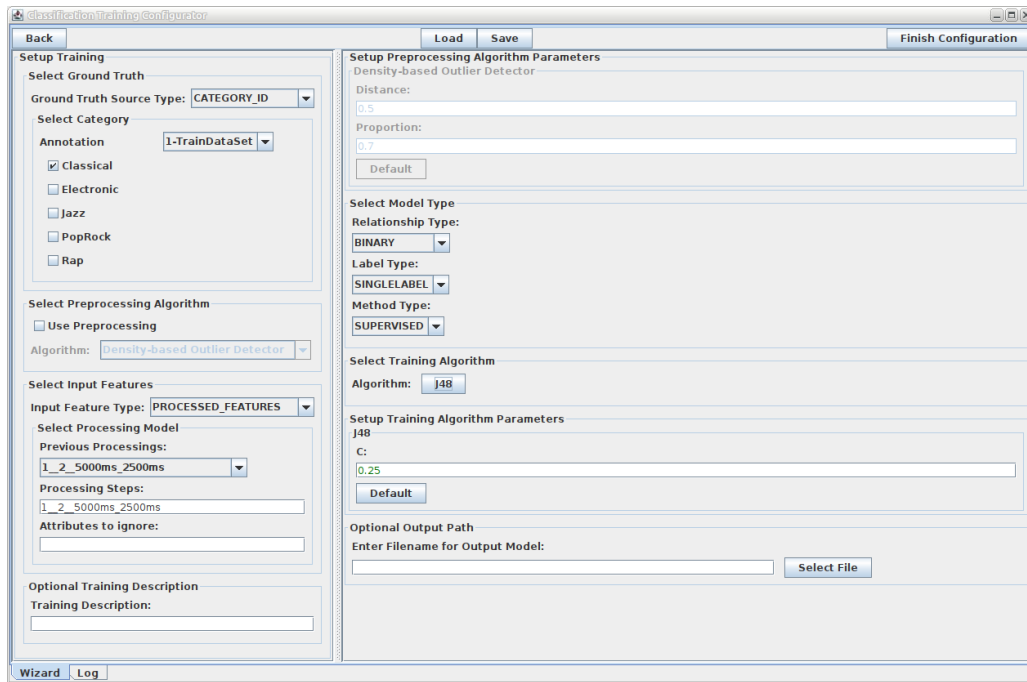


Figure 19: The Model Training Configurator.

For the training of supervised classification training methods, annotated data are required which can be selected in “Select Ground Truth” box. The different available annotations are stored in the file “multipleTracksAnnotationTable.arff” that is located in the config folder from AMUSE. Each ID refers to a file that contains a set of paths to tracks and their memberships to one or more categories. You can create this kind of annotation by using the Multiple Tracks Annotation Editor (see Section 4.4.2), or using a text editor. For every track, the processed feature file must be available in the Processed Feature Database. There are three different ways to specify the annotation at “Ground Truth Source Type”:

- **CATEGORY_ID**: An annotation from “multipleTracksAnnotationTable.arff”.
- **FILE_LIST**: An annotation that is chosen using a file chooser and does not have to be listed in “multipleTracksAnnotationTable.arff”.
- **READY_INPUT**: A path to ARFF file with ready features and annotations.

In the next step, you should decide whether you want to use preprocessing (like outlier removal) and configure it. This step is not necessary.

For the building of annotated feature vectors to train the model (unless READY_INPUT is pre-configured), the configuration of the feature processing steps has to be declared in the box “Select Processing Model”. By using this information, AMUSE can locate the processed features. The processing history saves the last processing descriptions. They have the following format:

- **PROCESSING STEPS**: At first, the chain of the processing methods is given, separated by a minus sign (‘-’). Method parameters are given in brackets and are separated by underscores (‘_’).
- **SEPARATOR**: Two underscores.
- **FEATURE MATRIX CONVERSION**: Method to build a harmonized feature matrix.
- **SEPARATOR**: Two underscores.
- **CLASSIFICATION WINDOW**: Length and step size for the classification windows.

An example is:

$$\underbrace{5[75]}_{\text{Processing steps}} \underbrace{--0[true.true]}_{\text{Conversion}} \underbrace{--5000ms_2500ms}_{\text{Classification window}}$$

Here, the processing is done with method 5 (principal components analysis) using 75% of components. The conversion method is GMM1 (the mean value and the standard deviation are saved). The classification window length is 5000 ms with 2500 ms step size.

Additionally, attributes to ignore that will not be used for training can be specified. Their IDs should be listed under “Attributes to ignore” (separated by commas).

Alternatively to processed features, also raw, unprocessed features can be given as training input. In order to do that, RAW_FEATURES has to be set as “Ground Truth Source Type”. When raw features are selected, you need to specify which features should be used and from which window and step size input feature vectors should be formed. We advice to only use raw features with neural networks from the Keras plugin, as large unprocessed and unstructured feature vectors are not really suitable for most other classification algorithms.

On the right side of the Classification Configurator, you can specify the model type in the box “Select model type”. The different model types are explained in Section 2.1.3. Selecting the model type limits the choice of training algorithm to those that support that model type. The list of supported model training algorithms is provided in Table 1.

Below, you can select the algorithm that should be used to train the classification model. Some of the algorithms have parameters that will appear on the right side of the training configuration window after selection. In the box “Optional Output Path”, you can provide an alternative storage place for the trained models (if it is left blank, AMUSE Model Database is used). Now, you can save your configuration.

4.3.4 Model Application Configurator

For the application of the previously trained models, the music tracks that should be classified need to be specified under “Select Classification Input”, as seen in Figure 20a. Similarly as in Section 4.3.3, they can be provided as an ARFF table (FILE_LIST), a category from the “multipleTracksAnnotationTable.arff” (CATEGORY_ID), or as ready features (READY_INPUT). If FILE_LIST or CATEGORY_ID are used, the processed features must be available in the Processed Feature Database for every track.

Next, the configurations from previous steps must be entered, as shown in Figure 20b, to allow AMUSE to locate the classification model and the processed features for the specified tracks. Additionally, you have to specify an ARFF file in which the results will be written.

4.3.5 Validation Configurator

The validation task is responsible for the evaluation and comparison of models.

In the first step, as shown in Figure 21, the test category and the training category have to be configured. Training category describes the ground truth used for evaluation—the unlabeled processed features must exist in the Processed Features Database. The model training algorithm and processing description must be set.

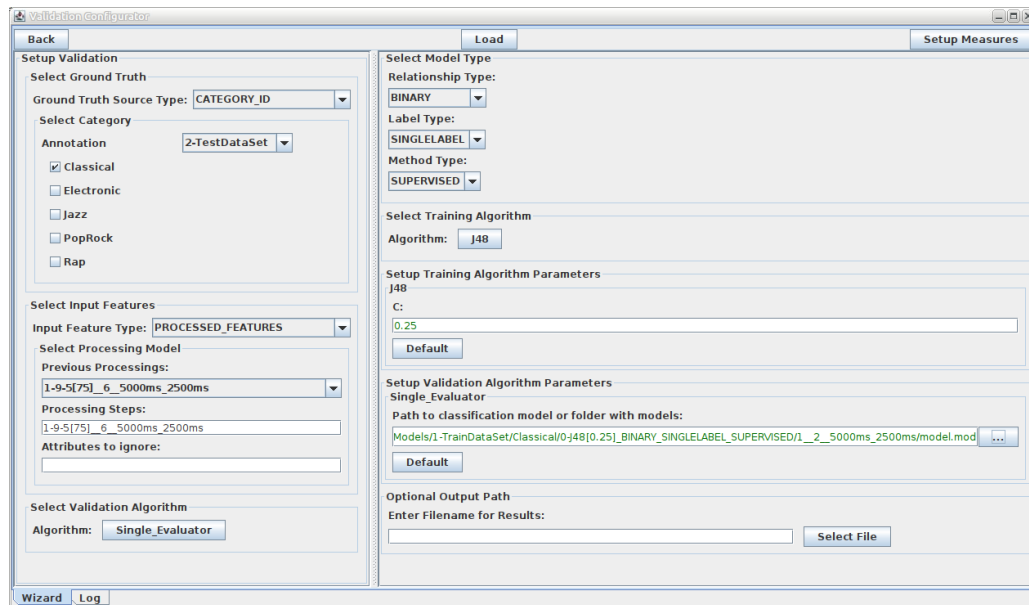
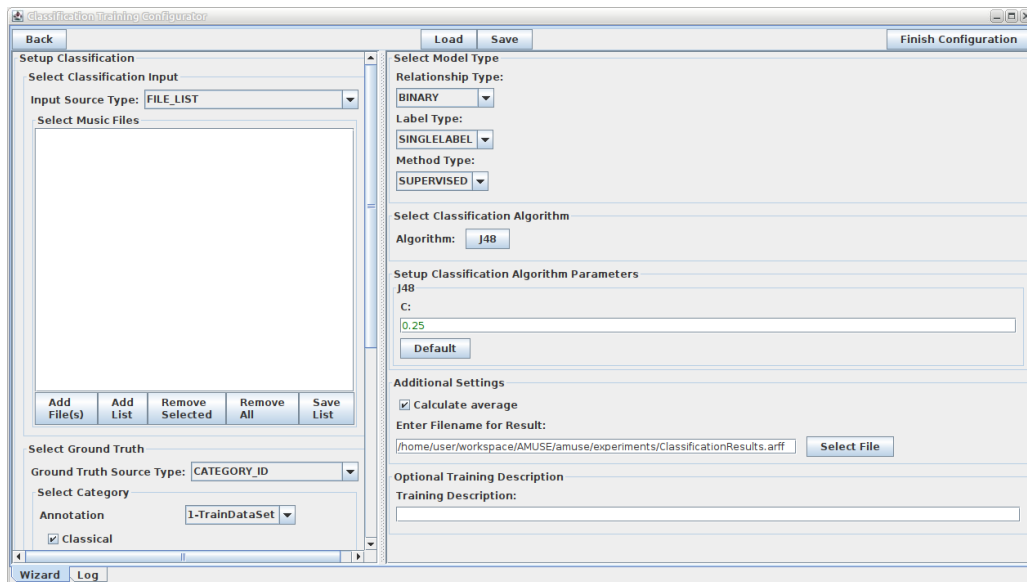
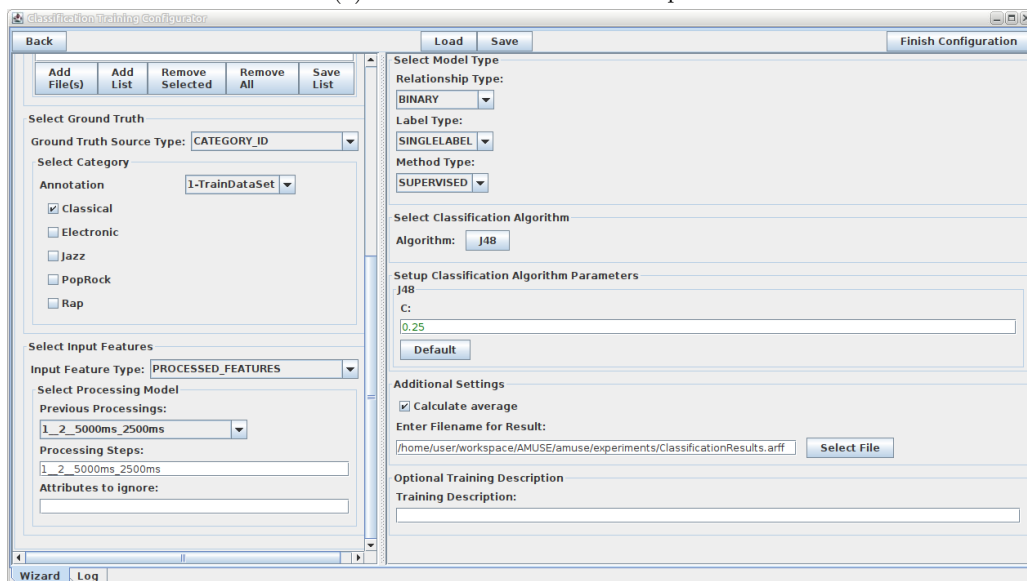


Figure 21: The Validation Configurator.

Then, you should decide whether you wish to validate a single previously trained model (“Single_Evaluator” method) and give a path to this model in the right panel, or to run an n -fold cross-validation where n models are trained automatically from the given category. In the second step, cf. Figure 22, you can enable/disable the measures which should be calculated. They will be stored in the AMUSE Measure Database folder.



(a) Selection of classification input.



(b) Selection of input features.

Figure 20: The Model Application Configurator.

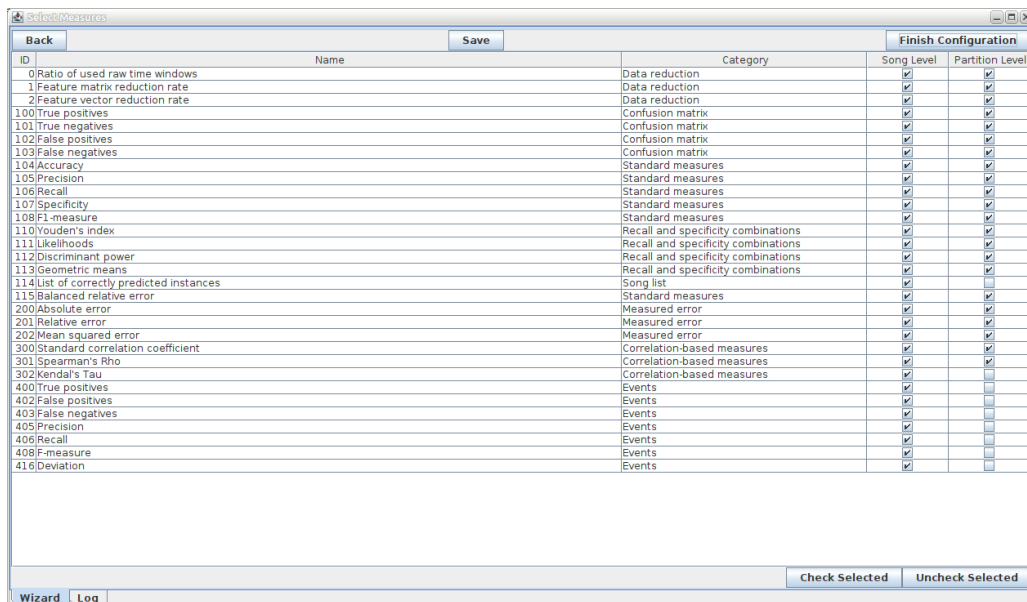


Figure 22: Selection of evaluation measures.

4.3.6 Optimization Configurator

Currently, the feature selection by evolutionary strategies (ES) is supported as an optimization method. During the optimization setup, you should select the category for training of the models using the features selected by ES. The second category is the optimization category for the model evaluation. The last one is the independent test category (is not used per default).

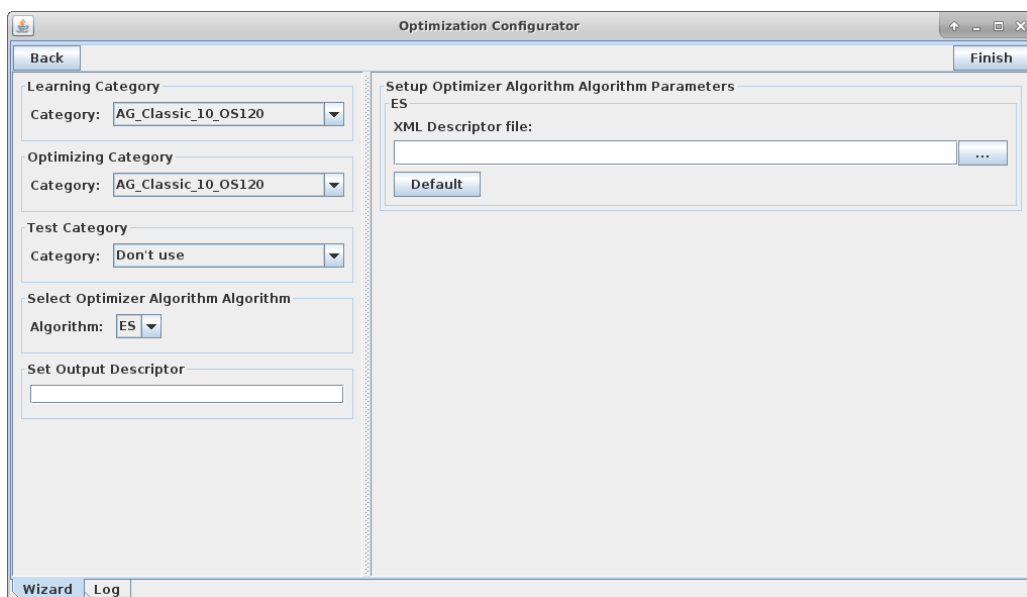


Figure 23: The Optimization Configurator.

The ES configuration is saved in an XML file, see `amuse\docs\optimization.xml` for an example. In this XML, the initial feature table, processing steps, and the measure table must be set. The first measure in the given measure list is used for optimization.

4.4 Annotation Editor

For supervised classification training, or also event detection, annotations are required. AMUSE features two different editors for the annotation process. The Single Track Annotation Editor is used to define annotations for an individual track (for example, temporal events or segments with different instrumentation). The Multiple Tracks Annotation Editor helps to annotate many tracks in a simple way, e.g., assigning them to genres or mood categories.

4.4.1 Single Track Annotation Editor

To start the Single Track Annotation Editor, please push the button “Create Single Track Annotation” in the AMUSE start screen. In the editor you can then define attribute values that may change in the course of the track. The data type of the attributes corresponds to AMUSE’s segment and event attributes. An example for the Single Track Annotation Editor is shown in Figure 24.

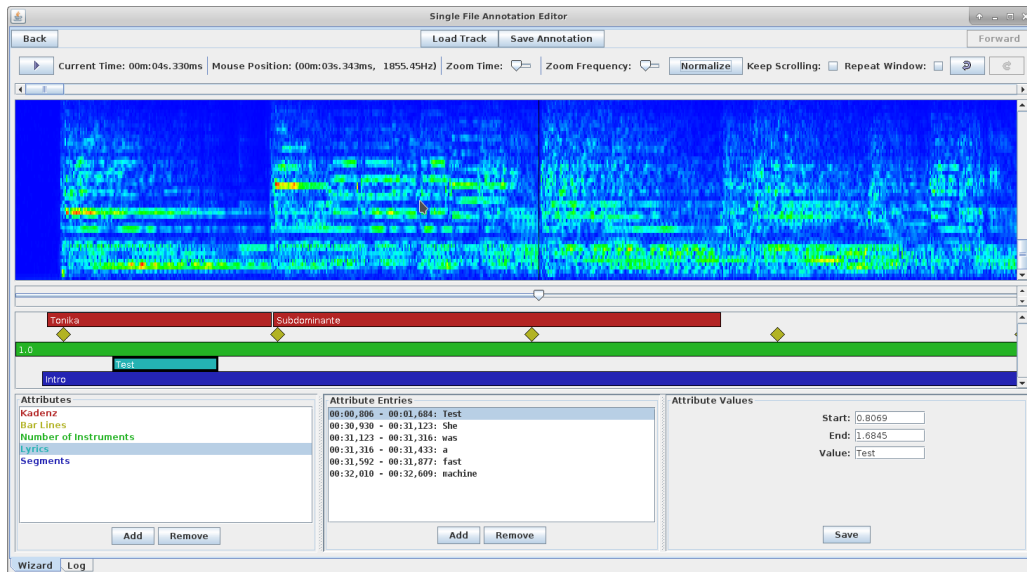


Figure 24: The Single Track Annotation Editor.

The following functionalities are available for music playback:

- **PLAY OR PAUSE THE MUSIC:** Press the Play/Pause button in the top left corner. Make sure that there is something left to play. Playing stops at the end of the track.
- **GO TO AN ARBITRARY TIME POINT IN THE MUSIC TRACK:** Click on the audio spectrum or the time slider beneath.
- **SCROLL AUTOMATICALLY WHILE THE MUSIC PLAYS:** Click on the checkbox right to “Keep Scrolling” in the top row. When checked, the editor will automatically scroll in such a way that the part of the audio spectrum is visible that corresponds to the currently played moment in time of the track.
- **REPEAT A SPECIFIED TIME WINDOW:** Click on the checkbox right to “Repeat Window”. The audio spectrum outside of the window is now darkened, as shown in an example in Figure 25.

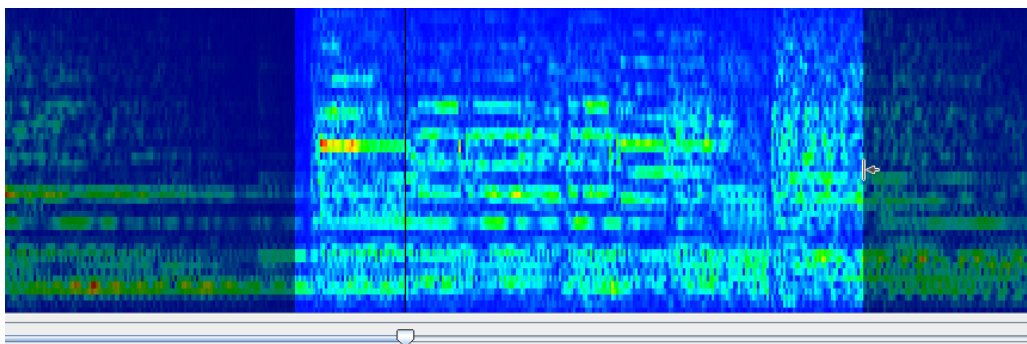


Figure 25: Changing the end of the window to be repeated.

To change the start and the end of the window, drag and drop the mouse, starting near the border. You can also move the whole window by starting the drag and drop inside it.

You can run the following operations on the audio spectrum:

- **SCALE THE AUDIO SPECTRUM:** The audio spectrum can be scaled along both the vertical and horizontal axis by shifting sliders. They can be found in the top row. “Zoom Time” corresponds to the horizontal and “Zoom Frequency” to the vertical axis.
- **NORMALIZE THE CURRENT WINDOW OF THE AUDIO SPECTRUM:** When the colors in the currently visible part of the audio spectrum are not easily distinguishable, the colors of this part can be normalized by using the “Normalize” button. Only this part is normalized. It persists as long as you do not scroll through the audio spectrum.
- **SCROLL THROUGH THE AUDIO SPECTRUM:** You can scroll through the audio spectrum by using the horizontal and vertical scrollbars around it. Also, you can drag and drop the audio spectrum to move to the desired part.

In the area below the spectrum, the annotations can be done:

- **ADD AN ATTRIBUTE:** Click on “Add” in the bottom left corner. A dialog appears, like shown in Figure 26.



Figure 26: The dialog for adding an attribute.

Here, you can select one or more attributes and add them by confirming the dialog.

- **IMPORT VALUES FROM AN EVENT FEATURE:** It is possible to import the values of an event feature previously extracted by AMUSE. To do so, select an event attribute (marked so in the column “Type”) in the dialog from Figure 26. Then, the button “Fill With Values From Amuse Event Feature” will be switched on.
- **REMOVE AN ATTRIBUTE:** Select an attribute in the list and click on “Remove” button below the list. You can either delete the file associated with the attribute or only hide the attribute. If you choose to only hide it, it will be shown again the next time the track is loaded.
- **ADD AN ATTRIBUTE ENTRY:** There are two ways to add an attribute entry. First, you should select the attribute in the list in the bottom left corner. Then, click on “Add” beneath the list for the attribute entries. An entry will be created, starting at the current position in the track. Apart from that, you can also add entries by interacting with the annotation visualization in the middle of the screen. There is a row for each attribute in which its entries are shown. Add an entry to it in an unoccupied space by either double-clicking for event attributes or dragging and dropping otherwise.
- **REMOVE AN ATTRIBUTE ENTRY:** Select the attribute entry in the annotation visualization or the list beneath. Then, click on “Remove” below the list for attribute entries.
- **ALTER THE START/ENDING TIME OF AN ATTRIBUTE ENTRY:** You can enter the wanted start and end times of a selected entry in the bottom right corner. Confirm with “Save”. Besides, you can drag and drop an entry’s box to the desired position in the annotation visualization. Apart from event attributes, it is also possible to change only the start or end by beginning to drag at the box’s start or end.
- **ALTER AN ATTRIBUTE ENTRY’S VALUE:** Select the attribute entry and enter the value in the bottom right corner. Afterwards, click on “Save”.
- **UNDO/REDO AN ACTION:** Click on the respective button in the top right corner.

Loading and Saving:

- **LOAD A DIFFERENT TRACK:** Click on “Load Track” in the top middle and select the track in the upcoming dialog.
- **SAVE AN ANNOTATION:** Click on “Save Annotation”. The files will be saved in the annotation database.
- **LOAD AN ANNOTATION:** Loading an annotation automatically takes place when loading a track, because the annotation files are saved in the annotation database.

4.4.2 Multiple Tracks Annotation Editor

To start the Multiple Tracks Annotation Editor, please push the button “Create Multiple Tracks Annotation” in the AMUSE start screen. Now, you can assign categories like musical genres or moods to multiple tracks. Figure 27 shows a screenshot of the editor.

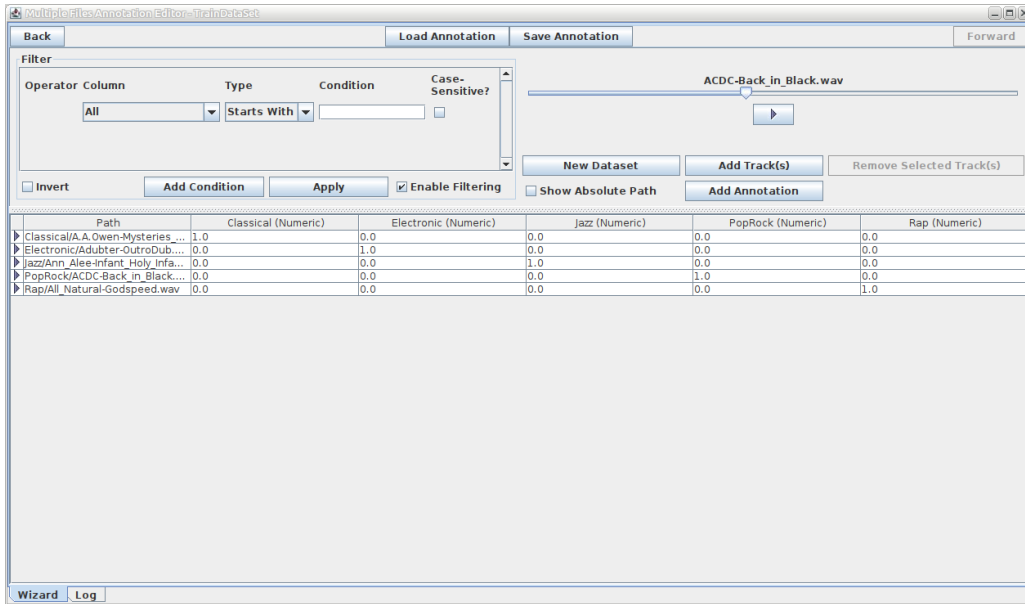


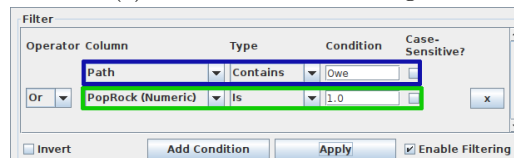
Figure 27: The Multiple Tracks Annotation Editor.

For a quick annotation, it is possible to apply filtering and get only rows with certain contents displayed while hiding the others. In the upper left corner, you can set up the filter, which should contain at least one condition. Every condition consists of a condition text, the configuration of how it should behave, and the name of the column, to whose contents the condition must apply. It is also possible to define a condition for all columns. There are four types of conditions: “Starts With”, “Ends With”, “Contains”, and “Is”. They determine in which way the condition text must occur in a row. By default, the condition text is case insensitive, which can be changed by ticking the checkbox in the column “Case-Sensitive?”. Multiple conditions can be used simultaneously. They are connected via the logical operators “And” and “Or”.

Figure 28 provides an example for filtering. The conditions “Path Contains Owe” and “PopRock Is 1.0” are displayed in Figure 28b. The first row from Figure 28a satisfies the first condition, the second satisfies none, the third also satisfies none, the fourth satisfies the second condition and the last satisfies none of the conditions, as implied by the colored boxes. Both conditions are connected with “Or”. When activated, only the rows that satisfy at least one of the conditions are displayed, as shown in Figure 28c.

Path	Classical (Numeric)	Electronic (Numeric)	Jazz (Numeric)	PopRock (Numeric)	Rap (Numeric)
Classical/A.A.Owen-Mysteries...	1.0	0.0	0.0	0.0	0.0
Electronic/Adubter-OutroDub...	0.0	1.0	0.0	0.0	0.0
Jazz/Ann_Alee-infant_Holy_Inf...	0.0	0.0	1.0	0.0	0.0
PopRock/ACDC-Back_in_Black...	0.0	0.0	0.0	1.0	0.0
Rap/All_Natural-Godspeed.wav	0.0	0.0	0.0	0.0	1.0

(a) The rows before filtering.



(b) The filter configuration.

Path	Classical (Numeric)	Electronic (Numeric)	Jazz (Numeric)	PopRock (Numeric)	Rap (Numeric)
Classical/A.A.Owen-Mysteries...	1.0	0.0	0.0	0.0	0.0
PopRock/ACDC-Back_in_Black...	0.0	0.0	0.0	1.0	0.0

(c) The rows after filtering.

Figure 28: Example for the filtering process

The following options are available for the filtering:

- **ENABLE/DISABLE FILTERING:** If you wish to disable filtering without having to delete all defined conditions, unmark the checkbox “Enable Filtering”.

- **INVERT THE FILTER:** To select all tracks for which a set of filter conditions is not fulfilled, check the box “Invert”.
- **ADD A CONDITION:** Click on the button “Add Condition”.
- **REMOVE A CONDITION:** Click on the button “X” to the right of the condition. The first condition cannot be removed.

During the annotation, you may do the following steps:

- **ADD TRACKS:** Click on the button “Add Track(s)”. In the upcoming dialog, you can make a selection of tracks and folders. When selecting a folder, it will be searched for tracks excluding its subfolders. By checking the box “Recursive?”, this search is done recursively in all subfolders.
- **REMOVE TRACKS:** Select the rows you want to remove and click on “Remove Selected Track(s)”.
- **ADD AN ANNOTATION (COLUMN):** You can add a column to annotate another property of the tracks by clicking on “Add Annotation” in the upper right corner. A dialog will prompt you to select, create, or remove a column.
- **REMOVE A COLUMN:** Right-click on the header of the column you want to delete. Click on “Delete” and confirm the dialog.
- **ENTER VALUES FOR ONE TRACK:** Select the row of the track for which you want to enter a value. Double-click or right click on the cell you want to edit.
- **ENTER VALUES FOR MULTIPLE TRACKS SIMULTANEOUSLY:** Select every row for which you want to enter a value. Double-click on the cell you want to edit in the last selected row or right-click in a column to enter a value for it. After applying, the value will be entered in each respective cell of every selected row.
- **PLAY MUSIC:** Click on the “Play” icon on the left of one row in the table. Now, you can control the playback in the top right corner.
- **SEARCH FOR A TIME POSITION IN A TRACK:** When playing music, click on the slider in the upper right corner at the desired position.
- **SAVE ANNOTATION:** An annotation can be saved as ARFF-file. Click on “Save Annotation” and enter the location it should be saved to.
- **LOAD ANNOTATION:** A previously saved annotation can be loaded by clicking on “Load Annotation”. If there are attributes that cannot be matched to the ones defined in AMUSE, you will be asked whether they should be omitted or matched to other attributes.

References

- [1] CHOLLET, F., ET AL. Keras. <https://keras.io>, 2021. Accessed on: 01.03.2021.
- [2] FRANK, E., HALL, M. A., HOLMES, G., KIRKBY, R., AND PFAHRINGER, B. WEKA - A machine learning workbench for data mining. In *The Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer, 2005, pp. 1305–1314.
- [3] GINSEL, P., VATOLKIN, I., AND RUDOLPH, G. Analysis of structural complexity features for music genre recognition. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)* (2020), IEEE, pp. 1–8.
- [4] GUYON, I., NIKRAVESH, M., GUNN, S., AND ZADEH, L. A., Eds. *Feature Extraction. Foundations and Applications*, vol. 207 of *Studies in Fuzziness and Soft Computing*. Springer, Berlin Heidelberg, 2006.
- [5] JETTE, M., DUNLAP, C., GARLICK, J., AND GRONDONA, M. Slurm: Simple linux utility for resource management, 2002.
- [6] JOLLIFFE, I. T. *Principal Component Analysis*. Springer, 2002.
- [7] MAUCH, M., AND LEVY, M. Structural change on multiple time scales as a correlate of musical complexity. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)* (2011), A. Klapuri and C. Leider, Eds., University of Miami, pp. 489–494.
- [8] MCENNIS, D., MCKAY, C., FUJINAGA, I., AND DEPALLE, P. jaudio: An feature extraction library. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)* (2005), pp. 600–603.
- [9] MCFEE, B., RAFFEL, C., LIANG, D., ELLIS, D. P. W., MCVICAR, M., BATTENBERG, E., AND NIETO, O. Librosa: Audio and music signal analysis in python. In *Proceedings the Python Science Conference* (2015), pp. 18–25.
- [10] MENG, A., AHRENDT, P., LARSEN, J., AND HANSEN, L. K. Temporal feature integration for music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing* 15, 5 (2007), 1654–1664.
- [11] MIERSWA, I., AND MORIK, K. Automatic feature extraction for classifying audio data. *Machine Learning Journal* 58, 2-3 (2005), 127–149.
- [12] PACHET, F., AND ZILS, A. Evolving automatically high-level music descriptors from acoustic signals. In *Proceedings of the 1st International Symposium on Computer Music Modeling and Retrieval (CMMR)* (2003), vol. 2771 of *Lecture Notes in Computer Science*, Springer, pp. 42–53.
- [13] RABINER, L., AND JUANG, B.-H. *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River, 1993.
- [14] VATOLKIN, I. Feature processing. In *Music Data Analysis: Foundations and Applications*, C. Weihs, D. Jannach, I. Vatulkin, and G. Rudolph, Eds. CRC Press, 2016, pp. 365–388.
- [15] VATOLKIN, I., GINSEL, P., AND RUDOLPH, G. Advancements in the music information retrieval framework amuse over the last decade. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)* (2021), pp. 2383–2389.
- [16] VATOLKIN, I., THEIMER, W., AND BOTTECK, M. AMUSE (Advanced MUSIC Explorer) - a multitool framework for music data analysis. In *Proceedings of the 11th International Society on Music Information Retrieval Conference (ISMIR)* (2010), J. S. Downie and R. C. Veltkamp, Eds., pp. 33–38.
- [17] VATOLKIN, I., AND WEIHS, C. Evaluation. In *Music Data Analysis: Foundations and Applications*, C. Weihs, D. Jannach, I. Vatulkin, and G. Rudolph, Eds. CRC Press, 2016, pp. 329–363.