gamma_flow: **G**uided **A**nalysis of **M**ulti-label spectra by **Ma**trix **F**actorization for **L**ightweight **O**perational **W**orkflows

Viola Rädle[a,*], Tilman Hartwig[a], Benjamin Oesen[a], Emily Alice Kröger[b], Julius Vogt[b], Eike Gericke[b], Martin Baron[b]

[a] *Application Lab for AI and Big Data, German Environmental Agency, Leipzig, Germany*
[b] *Federal Office for Radiation Protection, Berlin, Germany*

Abstract

gamma_flow is an open-source Python package for real-time analysis of spectral data. It supports classification, denoising, decomposition, and outlier detection of both single- and multi-component spectra. Instead of relying on large, computationally intensive models, it employs a novel supervised approach to non-negative matrix factorization (NMF) for dimensionality reduction. This ensures a fast, efficient, and adaptable analysis while reducing computational costs. **gamma_flow** achieves classification accuracies above 90% and enables reliable automated spectral interpretation. Originally developed for gamma-ray spectra, it is applicable to any type of one-dimensional spectral data. As an open and flexible alternative to proprietary software, it supports various applications in research and industry.

Keywords: Python, Gamma spectroscopy, Non-negative Matrix Factorization, Classification, Denoising, Spectral Deconvolution

Metadata

The ancillary data table 1 is required for the sub-version of the codebase. Please replace the italicized text in the right column with the correct information about your current code and leave the left column untouched.
Optionally, you can provide information about the current executable software version filling in the left column of Table 2. Please leave the first column as it is. FRAGE: Welche Tabelle sollen wir ausfüllen?

*Corresponding author: raedle.htwk@web.de

Nr.	Code metadata description	Metadata
TO DO C1	Current code version	For example v42
C2	Permanent link to code/repository used for this code version	`https://gitlab.opencode.de/uba-ki-lab/gamma_flow`
C3	Permanent link to Reproducible Capsule	TO DO For example: `https://codeocean.com/capsule/0270963/tree/v1`
C4	Legal Code License	BSD 3-Clause "New" or "Revised" License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python
C7	Compilation requirements, operating environments & dependencies	TO DO. Jupyter?
C8	If available Link to developer documentation/manual	`https://gitlab.opencode.de/uba-ki-lab/gamma_flow/-/blob/main/README.md?ref_type=heads`
C9	Support email for questions	raedle.htwk@web.de

Table 1: Code metadata (mandatory)

1. Motivation and significance

In this section, we want you to introduce the scientific background and the motivation for developing the software.

- *Explain why the software is important and describe the exact (scientific) problem(s) it solves.*
- *Indicate in what way the software has contributed (or will contribute in the future) to the process of scientific discovery; if available, please cite a research paper using the software.*
- *Provide a description of the experimental setting. (How does the user use the software?)*
- *Introduce related work in literature (cite or list algorithms used, other software etc.).*

Most radioactive sources can be identified by measuring their emitted radiation (X-rays and gamma rays), and visualizing them as a spectrum. In nuclear

Nr.	(Executable) software metadata description	Please fill in this column
S1	Current software version	For example 1.1, 2.4 etc.
S2	Permanent link to executables of this version	For example: `https://github.com/combogenomics/DuctApe/releases/tag/DuctApe-0.16.4`
S3	Permanent link to Reproducible Capsule	
S4	Legal Software License	List one of the approved licenses
S5	Computing platforms/Operating Systems	For example Android, BSD, iOS, Linux, OS X, Microsoft Windows, Unix-like , IBM z/OS, distributed/web based etc.
S6	Installation requirements & dependencies	
S7	If available, link to user manual - if formally published include a reference to the publication in the reference list	For example: `http://mozart.github.io/documentation/`
S8	Support email for questions	

Table 2: Software metadata (optional)

security applications, the resulting gamma spectra have to be analyzed in real-time as immediate reaction and decision making may be required. However, the manual recognition of isotopes present in a spectrum constitutes a strenous, error-prone task that depends upon expert knowledge. Hence, this raises the need for algorithms assisting in the initial categorization and recognizability of measured gamma spectra. The delineated use case brings along several requirements:

- As mobile, room temperature detectors are often deployed in nuclear security applications, the produced spectra typically exhibit a rather low energy resolution. In addition, a high temporal resolution is required (usually around one spectrum per second), leading to a low acquisition time and a low signal-to-noise ratio. Hence, the model must be robust and be able to handle noisy data.

- For some radioactive sources, acquisition of training spectra may be challenging. Instead, spectra of those isotopes are simulated using Monte Carlo N-Particle (MCNP) code [1]. In this process, energy

deposition in a detector material is simulated, yielding spectra that can be used for model training. However, simulated spectra and measured spectra from real-world sources may differ, which may be a constraint for model performance. On this account, preliminary data exploration is crucial to assess the similarity of spectral data from different detectors and to evaluate potential data limitations.

- Lastly, not only the correct classification of single-label test spectra (stemming from one isotope) is necessary, but also the decomposition of linear combinations of various isotopes (multi-label spectra). Hence, classification approaches like k-nearest-neighbours that solely depend on the similarity between training and test spectra are not applicable.

This paper presents **gamma_flow**, a Python package designed for the real-time analysis of one-dimensional spectra. It was developed in response to the practical challenges described above and supports the following tasks:

- classification of test spectra to identify their constituents,
- denoising to enhence visibility and reduce measurement noise,
- outlier detection to evaluate the model's applicability to test spectra.

Originally developed for gamma spectroscopy, **gamma_flow** is applicable to various domains including material science, chemistry, and environmental monitoring. It facilitates automated analysis in settings where fast interpretation of spectral data is essential. By integrating supervised dimensionality reduction with classical signal processing techniques, it extends prior work such as that of Bilton et al. [2], contributing to reproducible and interpretable spectral analysis pipelines.

2. Software description

Describe the software. Provide enough detail to help the reader understand its impact. **gamma_flow** is based on a dimensionality reduction model that constitutes a novel, supervised approach to non-negative matrix factorization (NMF). More explicitly, the spectral data matrix is decomposed into the product of two low-rank matrices denoted as the scores (spectral data in latent space) and the loadings (transformation matrix or latent components). The loadings matrix is predefined and consists of the mean spectra of the training isotopes. Hence, by design, the scores axes correspond to the share of an isotope in a spectrum, resulting in an interpretable latent space. As a result, the classification of a test spectrum can be read directly from its

(normalized) scores. In particular, shares of individual isotopes in a multi-label spectrum can be identified. This leads to an explainable quantitative prediction of the spectral constituents. The scores can be transformed back into spectral space by applying the inverse model. This inverse transformation rids the test spectrum of noise and results in a smooth, easily recognizable denoised spectrum. If a test spectrum of an isotope is unknown to the model (i.e. this isotope was not included in model training), it can still be projected into latent space. However, when the latent space information (scores) are decompressed, the resulting denoised spectrum does not resemble the original spectrum any more. Some original features may not be captured while new peaks may have been fabricated. This can be quantified by calculating the cosine similarity between the original and the denoised spectrum, which can serve as an indicator of a test spectrum to be an outlier.

2.1. Software architecture

Give a short overview of the overall software architecture; provide a pictorial overview where possible; for example, an image showing the components. If necessary, provide implementation details. Soll hier ein Schaubild erstellt werden von jupyter-files und auf welche python-files sie zugreifen? To do: Jore fragen, automatisches Schaubild erstellen in VSCode? Pycallgraph, mermaid,...

2.2. Software functionalities

Present the major functionalities of the software.
This python package consists of three jupyter notebooks that are executed consecutively. In this section, their functionality and is outlined, with an emphasis on the mathematical structure of the model.

2.2.1. Preprocessing and data exploration

The notebook `01_preprocessing.ipynb` synchronizes spectral data and provides a framework of visualizations for data exploration. All functions called in this notebook are found in `tools_preprocessing.py`.
During **preprocessing**, the following steps are performed:

- Spectral data files are converted from .xlsm/.spe data to .npy format and saved.
- Spectra of different energy calibrations are rebinned to a standard energy calibration.
- Spectral data are aggregated by label classes and detectors. Thus, it is possible to collect data from different files and formats.

- Optional: The spectra per isotope are limited to a maximum number.
- The preprocessed spectra are saved as .npy files.

Data exploration involves the following visualizations:

- For each label class (e.g. for each isotope), the mean spectra are calculated detector-wise and compared quantitatively by the cosine similarity.
- For each label class, example spectra are chosen randomly and plotted to provide an overview over the data.
- The cosine similarity is calculated and visualized as a matrix for all label classes and detectors.

This helps to assess whether the model can handle spectra from different detectors.

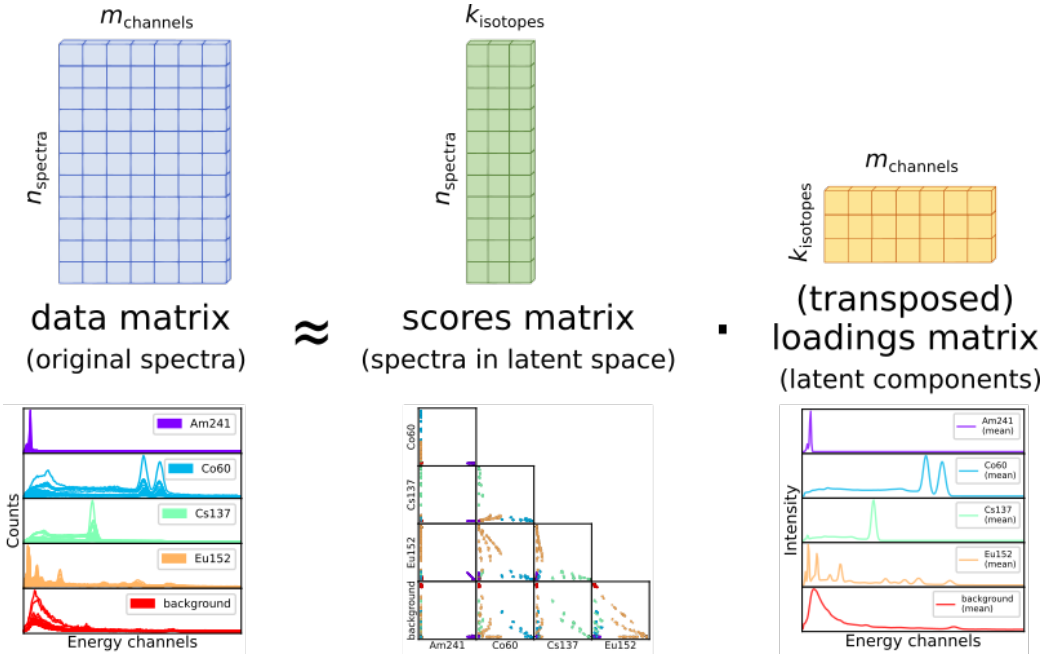
2.2.2. Model training and testing

The notebook `02_model.ipynb` trains and tests a dimensionality reduction model that allows for denoising, classification and outlier detection of test spectra. All functions called in this notebook are found in `tools_model.py`. The dimensionality reduction model presented in this paper comprises a matrix decomposition of spectral data. More precisely, the original spectra matrix X is reconstructed by two low-rank matrices S and L:

$$X \approx SL^T$$

with S: scores matrix (spectra in latent space)
L: loadings matrix (transformation matrix or latent components).

As illustrated in Figure 2.2.2, original spectral data can be compressed into k_{isotopes} dimensions. To ensure a conclusive assignment of the latent space axes to the isotopes (i.e. one axis stands for of one isotope), the loadings matrix is predefined as the mean spectra of the k_{isotopes} isotopes.
During model training, mean spectra for all isotopes are calculated. The scores are then derived by non-negative least squares fit of the original spectra to the loadings matrix. Thus, the components of the normalized scores vectors directly reveal the contributions of the individual isotopes. Denoised spectra, on the other hand, are computed by transforming the non-normalized scores back into spectral space (i.e. by multiplication of with the loadings matrix).



In mathematical terms, this model represents a 'supervised' approach to Non-negative Matrix Factorization (NMF) [3, 2]. While dimensionality reduction is conventionally an unsupervised task as it only considers data structure [4], our approach integrates labels in model training. This leads to an interpretable latent space and obviates the need for an additional classification step. While other supervised NMF approaches incorporate classification loss in model training [5, 6, 7], our model focuses on a comprehensible construction of the latent space.
The model is trained using spectral data from the specified detectors `dets_tr` and isotopes `isotopes_tr`. Subsequently, it is inferenced (i.e. scores are calculated) on three different test datasets:

1. validation data/holdout data from same detector as used in training (each spectrum including only one isotope or pure background)
2. test data from different detector (each spectrum including one isotope and background)
3. multi-label test data from different detector (each spectrum including multiple isotopes and background)

For all test datasets, spectra are classified and denoised. The results are visualized as

- confusion matrix

- misclassified spectra
- denoised example spectrum
- misclassification statistics
- scores as scatter matrix
- mean scores as bar plot

This helps to assess model performance with respect to classification and denoising.

2.2.3. Outlier analysis

The notebook `03_outlier.ipynb` provides an exploratory approach to outliers detection, i.e. to identify spectra from isotopes that were not used in model training. All functions called in this notebook are found in `tools_outlier.py`. To simulate outlier spectra, a mock dataset is generated by training a model after removing one specific isotope. The trained model is then inferenced on spectra of this unknown isotope to investigate its behaviour with outliers. First, the resulting latent space distribution and further meta data are analyzed to distinguish known from unknown spectra. Using a decision tree, the most informative feature is identified. Next, a decision boundary is derived for this feature, by
a) using the condition of the first split in the decision tree
b) fitting a logistic regression (sigmoid function) to the data
c) setting a manual threshold by considering accuracy, precision and recall of outlier identification.
The derived decision boundary can then be implemented in the measurement pipeline by the user.

Apart from the jupyter notebooks and python files described above, the project includes the following python files:

- `globals.py`: global variables
- `plotting.py`: all visualizations and plotting routines
- `util.py`: basic functions that are used by all notebooks

2.3. Sample code snippets analysis (optional)

Würde ich eher weglassen? Ggf. Das Modell vorstellen, aber eher nicht

3. Illustrative examples

Provide at least one illustrative example to demonstrate the major functions of your software/code.
***Optional**: you may include one explanatory video or screencast that will appear next to your article, in the right hand side panel. Please upload any video as a single supplementary file with your article. Only one MP4 formatted, with 150MB maximum size, video is possible per article. Recommended video dimensions are 640 x 480 at a maximum of 30 frames / second. Prior to submission please test and validate your .mp4 file at* `http://elsevier-apps.sciverse.com/GadgetVideoPodcastPlayerWeb/verification` *. This tool will display your video exactly in the same way as it will appear on ScienceDirect.*

Plots:

- Confusion matrix (different detector, single-label class.)
- Denoised spectrum
- ggf. Outlier: Feature importance (decision tree)

4. Impact

This is the main section of the article and reviewers will weight it appropriately. Please indicate:

- *Any new research questions that can be pursued as a result of your software.*
- *In what way, and to what extent, your software improves the pursuit of existing research questions.*
- *Any ways in which your software has changed the daily practice of its users.*
- *How widespread the use of the software is within and outside the intended user group (downloads, number of users if your software is a service, citable publications, etc.).*
- *How the software is being used in commercial settings and/or how it has led to the creation of spin-off companies.*

Please note that points 1 and 2 are best demonstrated by references to citable publications.

übernommen aus JOSS-Statement of need: In many research fields, spectral measurements help to assess material properties. In this context, an area of interest for many researchers is the classification (automated labelling) of the measured spectra. Proprietary spectral analysis software, however, are often limited in their functionality and adaptability [8, 9]. In addition, the underlying mechanisms are usually not revealed and may act as a black-box system to the user [10]. On top of that, a spectral comparison is typically only possible for spectra of pure substances [11]. However, there may be a need to decompound multi-label spectra (linear combinations of different substances) and identify their constituents.

gamma_flow is a Python package that can assist researchers in the classification, denoising and outlier detection of spectra. It includes data preprocessing, data exploration, model training and testing as well as an exploratory section on outlier detection. Making use of matrix decomposition methods, the designed model is lean and performant. Training and inference do not require special hardware or extensive computational power. This allows real-time application on ordinary laboratory computers and easy implementation into the measurement routine. The provided example dataset contains gamma spectra of several measured and simulated isotopes as well as pure background spectra. While this package was developed in need of an analysis tool for gamma spectra, it is suitable for any one-dimensional spectra. Examplary applications encompass

- **Infrared spectroscopy** for the assessment of the polymer composition of microplastics in water [12, 13]
- **mass spectrometry** for protein identification in snake venom [14, 15]
- **Raman spectroscopy** for analysis of complex pharmaceutical mixtures and detection of dilution products like lactose [16]
- **UV-Vis spectroscopy** for detection of pesticides in surface waters [17, 18]
- **stellar spectroscopy** to infer the chemical composition of stars [19]

5. Conclusions

... to do

Acknowledgements

We gratefully acknowledge the support provided by the Federal Ministry for the Environment, Nature Conservation and Nuclear Safety (BMUV), whose funding has been instrumental in enabling us to achieve our research objectives and explore new directions. We also extend our appreciation to Martin Bussick in his function as the AI coordinator. Additionally, we thank the entire AI-Lab team for their support and inspiration, with special recognition to Ruth Brodte for guidance on legal and licensing matters.

References

[1] J. A. Kulesza, T. R. Adams, J. C. Armstrong, S. R. Bolding, F. B. Brown, J. S. Bull, T. P. Burke, A. R. Clark, R. A. A. Forster III, J. F. Giron, T. S. Grieve, C. J. Josey, R. L. Martz, G. W. McKinney, E. J. Pearson, M. E. Rising, C. J. C. Solomon Jr., S. Swaminarayan, T. J. Trahan, S. C. Wilson, A. J. Zukaitis, MCNP® Code Version 6.3.0 Theory & User Manual, Tech. Rep. LA-UR-22-30006, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (Sep. 2022). doi:10.2172/1889957.

[2] K. J. Bilton, T. H. Joshi, M. S. Bandstra, J. C. Curtis, B. J. Quiter, R. J. Cooper, K. Vetter, Non-negative Matrix Factorization of Gamma-Ray Spectra for Background Modeling, Detection, and Source Identification, IEEE Transactions on Nuclear Science 66 (5) (2019) 827–837. doi:10.1109/TNS.2019.2907267.

[3] P. Shreeves, Dimensionality reduction techniques with applications in Raman spectroscopy - UBC Library Open Collections, Ph.D. thesis, University of British Columbia (2020).

[4] J. Olaya, C. Otman, Non-negative Matrix Factorization for Dimensionality Reduction, ITM Web of Conferences 48 (2022) 03006. doi:10.1051/itmconf/20224803006.

[5] J. Leuschner, M. Schmidt, P. Fernsel, D. Lachmund, T. Boskamp, P. Maass, Supervised non-negative matrix factorization methods for MALDI imaging applications, Bioinformatics 35 (11) (2019) 1940–1947. doi:10.1093/bioinformatics/bty909.

[6] H. Lee, J. Yoo, S. Choi, Semi-Supervised Nonnegative Matrix Factorization, IEEE Signal Processing Letters 17 (1) (2010) 4–7. doi:10.1109/LSP.2009.2027163.

[7] V. Bisot, R. Serizel, S. Essid, G. Richard, Supervised nonnegative matrix factorization for acoustic scene classification, in: IEEE International Evaluation Campaign on Detection and Classification of Acousitc Scenes and Events (DCASE 2016), Budapest, Hungary, 2016.

[8] H. Lam, Building and Searching Tandem Mass Spectral Libraries for Peptide Identification, Molecular & Cellular Proteomics 10 (12) (Dec. 2011). doi:10.1074/mcp.R111.008565.

[9] J. Nasereddin, M. Shakib, Ira: A free and open-source Fourier transform infrared (FTIR) data analysis widget for pharmaceutical applications, Analytical Letters 56 (16) (2023) 2637–2648. doi:10.1080/00032719.2023.2180516.

[10] L. El Amri, A. Chetaine, H. Amsil, B. El Mokhtari, H. Bounouira, A. Didi, A. Benchrif, K. Laraki, H. Marah, New open-source software for gamma-ray spectra analysis, Applied Radiation and Isotopes 185 (2022) 110227. doi:10.1016/j.apradiso.2022.110227.

[11] W. Cowger, Z. Steinmetz, A. Gray, K. Munno, J. Lynch, H. Hapich, S. Primpke, H. De Frond, C. Rochman, O. Herodotou, Microplastic Spectral Classification Needs an Open Source Community: Open Specy to the Rescue!, Analytical Chemistry 93 (21) (2021) 7543–7548. doi:10.1021/acs.analchem.1c00123.

[12] B. Ferreiro, R. Leardi, E. Farinini, J. M. Andrade, Supervised classification combined with genetic algorithm variable selection for a fast identification of polymeric microdebris using infrared reflectance, Marine Pollution Bulletin 195 (2023) 115540. doi:10.1016/j.marpolbul.2023.115540.

[13] Q. T. Whiting, K. F. O'Connor, P. M. Potter, S. R. Al-Abed, A high-throughput, automated technique for microplastics detection, quantification, and characterization in surface waters using laser direct infrared spectroscopy, Analytical and Bioanalytical Chemistry 414 (29) (2022) 8353–8364. doi:10.1007/s00216-022-04371-2.

[14] A. Zelanis, D. A. Silva, E. S. Kitano, T. Liberato, I. Fukushima, S. M. T. Serrano, A. K. Tashima, A first step towards building spectral libraries as complementary tools for snake venom proteome/peptidome studies, Comparative Biochemistry and Physiology Part D: Genomics and Proteomics 31 (2019) 100599. doi:10.1016/j.cbd.2019.100599.

[15] N. Ç. Yasemin, A. Izzet, K. M. Ali, K. Selçuk, S. Bekir, Identification of Snake Venoms According to their Protein Content Using the MALDI-TOF-MS Method, Analytical Chemistry Letters 11 (2) (2021) 153–167. doi:10.1080/22297928.2021.1894974.

[16] X. Fu, L.-m. Zhong, Y.-b. Cao, H. Chen, F. Lu, Quantitative analysis of excipient dominated drug formulations by Raman spectroscopy combined with deep learning, Analytical Methods 13 (1) (2021) 64–68. doi:10.1039/D0AY01874K.

[17] Y. Guo, C. Liu, R. Ye, Q. Duan, Advances on Water Quality Detection by UV-Vis Spectroscopy, Applied Sciences 10 (19) (2020) 6874. doi:10.3390/app10196874.

[18] X. Qi, Y. Lian, L. Xie, Y. Wang, Z. Lu, Water quality detection based on UV-Vis and NIR spectroscopy: A review, Applied Spectroscopy Reviews 59 (8) (2024) 1036–1060. doi:10.1080/05704928.2023.2294458.

[19] D. F. Gray, The Observation and Analysis of Stellar Photospheres, https://www.cambridge.org/highereducation/books/the-observation-and-analysis-of-stellar-photospheres/67B340445C56F4421BCBA0AFFAAFDEE0 (Dec. 2021). doi:10.1017/9781009082136.

References

[1] Use this style of ordering. References in-text should also use a similar style.

If the software repository you used supplied a DOI or another Persistent IDentifier (PID), please add a reference for your software here. For more guidance on software citation, please see our guide for authors or this article on the essentials of software citation by FORCE 11, of which Elsevier is a member.

Reminder: Before you submit, please delete all the instructions in this document, including this paragraph. Thank you!