GAMMA_FLOW: Guided Analysis of Multi-label spectra by Matrix Factorization for Lightweight Operational Workflows

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
PACS: 07.05.Kf, 29.30.Kv, 02.50.Sk
2020 MSC: 15A23, 62H25, 65D10

Metadata

1. Motivation and significance

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

*Corresponding author: raedle.htwk - AT - web.de

Nr.	Code metadata description	Metadata
TO DO C1	Current code version	0.9.0
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	-
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	Python >= 3.12, matplotlib, numpy, pandas, scikit_learn, scipy, seaborn, openpyxl
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 - AT - web.de

Table 1: Code metadata (mandatory)

In nuclear 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 on 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

Nr.	(Executable) software metadata description	Please fill in this column
S1	Current software version	0.9.0
S2	Permanent link to executables of this version	`https://gitlab.opencode.de/uba-ki-lab/gamma_flow`
S3	Permanent link to Reproducible Capsule	-
S4	Legal Software License	BSD 3-Clause "New" or "Revised" License
S5	Computing platforms/Operating Systems	Linux, Microsoft Windows
S6	Installation requirements & dependencies	Python >= 3.12, matplotlib, numpy, pandas, scikit_learn, scipy, seaborn, openpyxl
S7	If available, link to user manual - if formally published include a reference to the publication in the reference list	`https://gitlab.opencode.de/uba-ki-lab/gamma_flow/-/tree/main/documentation?ref_type=heads`
S8	Support email for questions	raedle.htwk - AT - web.de

Table 2: Software metadata (optional)

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 unknown 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

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.
For spectral denoising, the scores are 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

GAMMA_FLOW consists of three jupyter notebooks that are executed consecutively, as depicted in Figure 1. Each imports functions from a designated Python file, e.g. all functions called in `02_model.ipynb` are found in `tools_model.py`. In addition, the Python files `util.py`, `globals.py` and `plotting.py` provide elementary for all three notebooks.

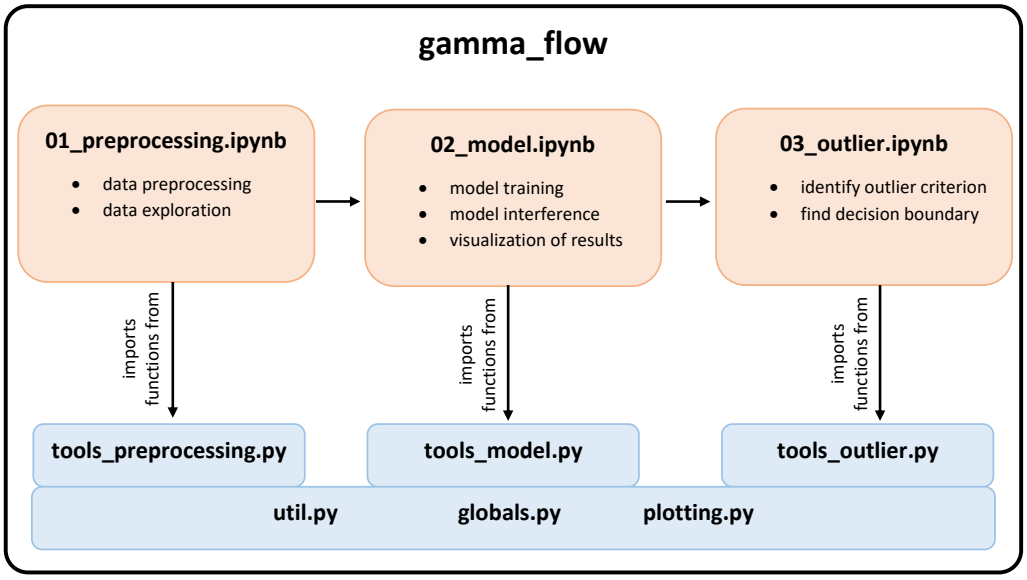

Figure 1: Software architecture of GAMMA_FLOW: The jupyter notebooks `01_preprocessing.ipynb`, `02_model.ipynb` and `03_outlier.ipynb` are executed consecutively, using functions from different Python files.

2.2. Software functionalities

In this section, the functionality of the software is outlined, with an emphasis on the mathematical structure of the model. To this end, the procedures realised in the three jupyter notebooks `01_preprocessing.ipynb`, `02_model.ipynb` and `03_outlier.ipynb` are delineated. In the software repository, a sample dataset of gamma spectra with examplary results is provided to give users a first insight into the software.

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 (for class balance).
- 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 model presented in this paper comprises a matrix decomposition of spectral data for dimensionality reduction. 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, 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. It represents

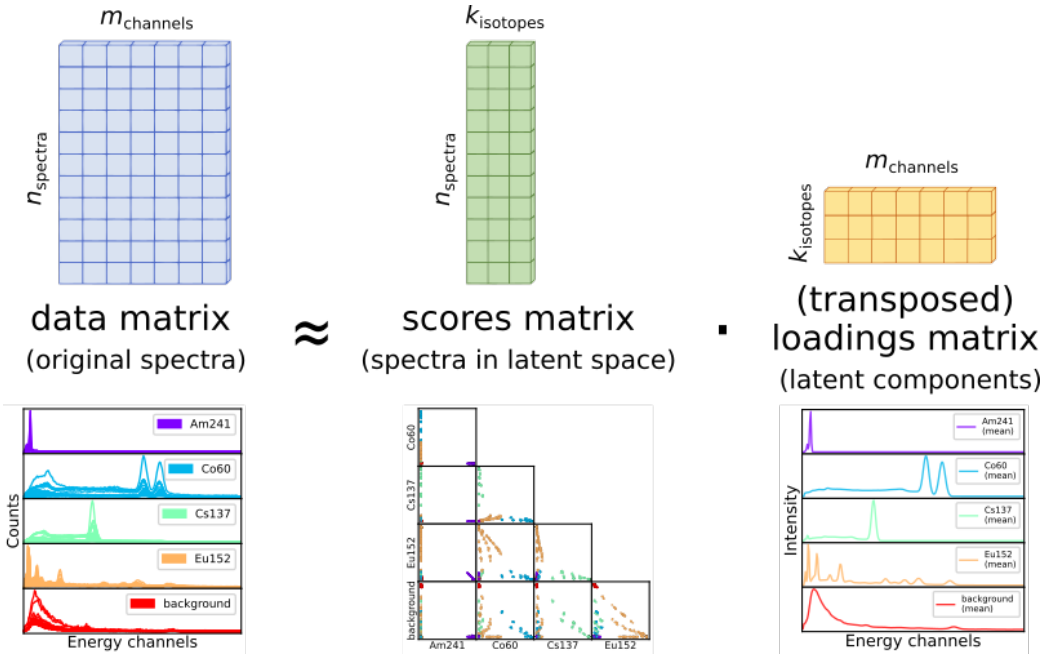

Figure 2: Decomposition of spectral data into scores and loadings, illustrated for five different isotopes. The loadings matrix contains the latent components, corresponding to the mean spectrum of each isotope. The scores matrix provides the representation of each spectrum in the five-dimensional latent space. When normalized, the scores indicate the predicted contribution of each isotope to the measured spectrum.

the latent components and serves as transformation matrix between spectral space and latent space.
In mathematical terms, this model represents a supervised approach to Non-negative Matrix Factorization [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. In constrast to other supervised NMF approaches that incorporate classification loss in model training [5, 6, 7], our model focuses on a comprehensible construction of the latent space.

During model training, mean spectra for all isotopes are calculated. The scores are then derived by a non-negative least squares fit of the original spectra to the loadings matrix. This enables the **spectral decomposition** and **classification** of the spectra, as 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 with the loadings matrix).

To assess **model performance**, the model is trained using spectral data from the specified detectors `dets_tr` and isotopes `isotopes_tr` and 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

3. Illustrative example

The major functions of `gamma_flow` include classification, denoising, decomposition, and outlier detection of spectra. In this section, these capabilities are demonstrated using an illustrative example.

A model is trained based on spectra of the isotopes Americium-241 (^{241}Am), Cobalt-60 (^{60}Co), Caesium-137 (^{137}Cs), Europium-152 (^{152}Eu), and pure background. The spectra are acquired using detector `det_tr`, and the isotope spectra are free of background radiation.

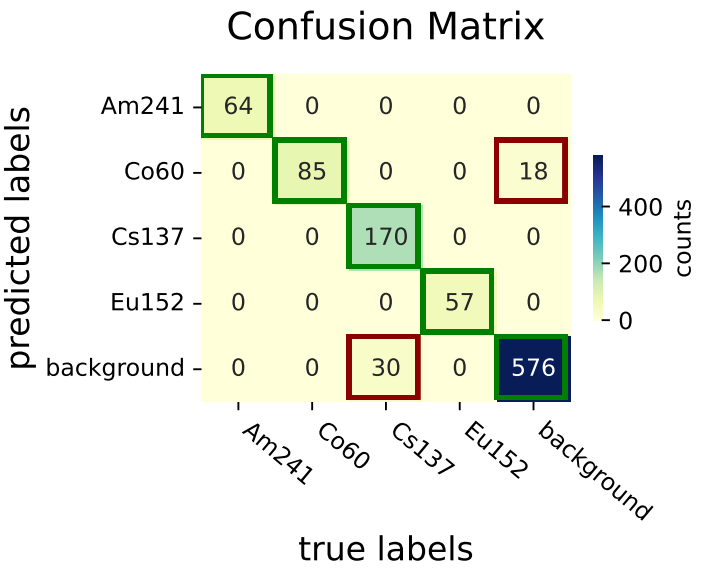

Figure 3: Confusion matrix of single-label classification, with an accuracy of 94.8%. While the model was trained on spectra from detector `det_tr`, spectra from a different detector were used as test data. Misclassifications only occur between an isotope and background, not between different isotopes.

The model is then inferenced on various test datasets to assess its performance in different scenarios (for more detail, see Section 2.2.2). The following example focuses on test dataset 2, in which test spectra were recorded using a different detector. Each test spectrum contains either a single isotope combined with background or pure background only; no background subtraction was applied.

The **classification** results for this test dataset can be seen in Figure 3, where the predicted labels are plotted against the ground truth as a confusion matrix. Correct classifications are highlighted with green borders while misclassifications are framed in red. Despite the difference between training and test spectra (different detector and presence of background in test data), an overall accuracy of 94.8% is achieved. Notably, all misclassifications involve confusion between an isotope and pure background — never between two isotopes. Such cases typically occur when the isotope signal is near the detection threshold and the model's prediction deviates from the manually assigned label. Please refer to the software documentation in the code repository for further visualizations on model training and classification results, such as the loadings, the scores (visualized as scatter matrix), the mean predicted scores by isotope, misclassified example spectra and misclassification statistics.

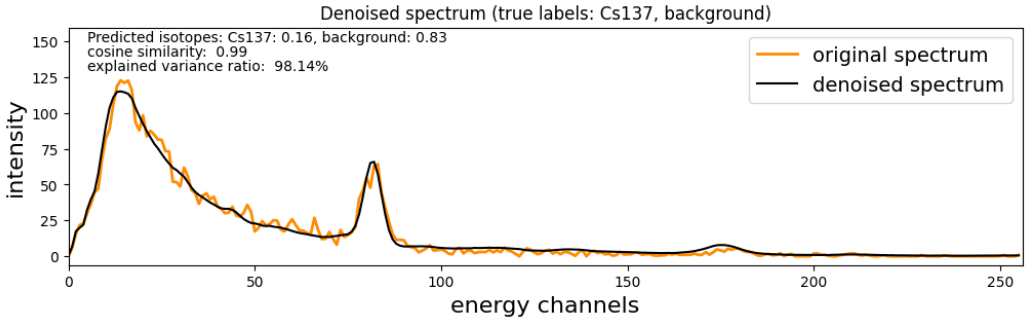

Figure 4: Original and denoised spectrum of ^{137}Cs and background. The denoised spectrum is smoother, leading to an easy and fast interpretability.

As described in Section 2, test spectra are **denoised** by transforming the scores back into spectral space by multiplication with the loadings matrix. The resulting spectra are significantly smoother and easier to interpret. A denoised example spectrum of ^{137}Cs and background is depicted in Figure 4. As can be seen in the annotations, the predictions are correct, with an additional information on the contribution of the components: The model predicts 16% share of ^{137}Cs and 83% background. The reconstruction of the original spectrum by the denoised spectrum is quantified by their cosine similarity of 0.99 and the explained variance ration of 98.14%, which both indicate a successfull denoising process.

The options for **outlier detection** are analyzed as described in Section 2.2.3. After the most informative feature is determined (in our case the cosine similarity between original and denoised spectrum) a decision boundary is derived. This can be done manually by considering accuracy, precision and recall, as visualized in Figure 5. Those quantities yield different information on the separation of known and unknown spectra:

- **Accuracy** quantifies the correct classifications as known and unknown spectra: $\frac{\text{True Positives+True Negatives}}{\text{True Positives+False Positives+True Negatives+False Negatives}}$
- **Precision** quantifies how many of the predicted outliers are actually unknown spectra: $\frac{\text{True Positives}}{\text{True Positives+False Positives}}$
- **Recall** quantifies how many of the unknown spectra are found: $\frac{\text{True Positives}}{\text{True Positives+False Negatives}}$

In this example, a threshold between 0.5 and 0.9 would be an adequate choice.

Figure 5: For outlier detection, the decision boundary can be derived based on accuracy, precision and recall of the outliers dataset.

4. Impact

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

In this paper, we introduced GAMMA_FLOW, an open-source, Python-based framework for the analysis of one-dimensional spectra. The approach combines established data science techniques in a resource-efficient way, enabling real-time analysis at low computational cost. Based on a supervised variant of non-negative matrix factorization, it constructs a fast and interpretable model for classification, decomposition, denoising, and outlier detection of spectral data. The method is applicable to both pure (single-label) and mixed (multi-label) spectra.
For a user-friendly implementation, the software is organized into three jupyter notebooks executed sequentially. Each notebook combines explanatory text with code, guiding users step-by-step through data preprocessing and exploration, model training and evaluation, and outlier detection. This enables users not only to apply the model to spectral data but also develop a deep understanding of the underlying processes - without requiring advanced mathematical or programming expertise.
GAMMA_FLOW bridges the gap between complex, often proprietary spectral analysis tools and the limitations of manual interpretation in laboratory workflows. By offering an open, transparent, and extensible framework, it empowers researchers to move beyond traditional black-box approaches. With its modular architecture and general applicability, it opens new avenues for innovation across disciplines — from nuclear safety and materials science to environmental monitoring and beyond. As such, GAMMA_FLOW has the potential to accelerate scientific discovery wherever spectral data plays a role.

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.