nature portfolio

Peer Review File

Multi-modal learning with incomplete data

Corresponding Author: Professor Tero Aittokallio

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
This manuscript presents iMML, a Python package for multi-modal learning with incomplete data, integrating more than 25 existing algorithms and providing unified interfaces for preprocessing, imputation, clustering, classification, and analysis. The software is well documented, maintained, and reproducible, and the provided tutorials demonstrate its usability across several domains.
The work addresses an important practical gap in the field, namely the lack of standardized tools for incomplete multi-modal learning.

Strengths
1. Strong practical relevance
Handling incomplete multi-modal data is a common and challenging problem across many domains such as healthcare, bioinformatics, and social sciences. By providing a unified implementation of a wide range of existing methods, the proposed package significantly lowers the barrier for practitioners and researchers working in this area.

2. Comprehensive and well-integrated framework
The library integrates more than 25 algorithms covering multiple tasks and methodological families. The unified interface for preprocessing, simulation of missingness, model training, and evaluation makes the framework convenient for both experimentation and applied use.

3. High-quality software engineering
The repository is well structured, actively maintained, and easy to install and run. The documentation, tutorials, and example workflows are clear and helpful. The code appears robust and reproducible, making it a useful resource for the community.

4. Emphasis on reproducibility and usability
The inclusion of simulation tools, standardized workflows, and compatibility with common Python ecosystems (NumPy, PyTorch, scikit-learn) greatly enhances the usability of the package and supports reproducible research.

Suggestions for Improvement

1. Evaluation is illustrative rather than systematic
The experimental case studies effectively demonstrate the usage of the package and illustrate its flexibility. However, the current evaluation appears primarily demonstrative rather than systematic.
It would be helpful to clarify whether the presented results are intended as usage examples rather than comprehensive benchmarking. If possible, adding a more structured comparison (e.g., standardized datasets, consistent settings across multiple methods) would further strengthen the empirical support.

2. Missingness assumptions and robustness claims could be clarified
The manuscript highlights the capability of the framework to handle different missingness scenarios. However, many of the integrated methods were originally developed under specific assumptions (e.g., MCAR or MAR). It would be beneficial to clarify:
1. Under what assumptions the included methods are expected to perform reliably

2. Whether robustness to different missingness mechanisms has been systematically evaluated
3. The extent to which the framework itself, versus the individual methods, provides robustness

(Remarks on code availability)
The repository is publicly available and well organized. The project structure is clear, with modular components for data preprocessing, missingness simulation, model implementation, and evaluation. The code follows a consistent design pattern and integrates well with common Python ecosystems such as NumPy, PyTorch, and scikit-learn.
The package provides comprehensive documentation through an online documentation site, including an overview of the framework, installation instructions, API references, and multiple usage tutorials. The README and documentation contain sufficient guidance for setting up the environment and running the main functionalities. The provided examples and notebooks help users understand typical workflows and reproduce the experiments presented in the manuscript.
The installation process is straightforward, and the core functionalities can be executed without major issues. The implementation appears stable and actively maintained. The modular design also makes it relatively easy for users to extend the framework or integrate their own methods.
Regarding reproducibility, the available scripts and examples allow users to reproduce the main experimental pipelines described in the paper at a functional level. While some experimental settings (e.g., dataset-specific preprocessing details or hyperparameter configurations) could be further specified for exact numerical reproduction, the current resources are sufficient to reproduce the methodology and overall results.
Overall, the code is well documented, usable, and represents a valuable resource for the community.

Reviewer #2

(Remarks to the Author)
Summary
This manuscript proposes iMML, an open-source Python package designed to fill the gap in the multi-modal learning community: handling incomplete data in a unified, user-friendly framework. iMML integrates over 25 published algorithms for tasks such as imputation, clustering, classification, retrieval, feature extraction/selection, and visual exploration. The authors have implemented most methods in Python, wrapped the remaining Matlab/R code for seamless use, and provided a consistent scikit-learn–style API, extensive documentation, and high coverage unit testing across platforms. Six illustrative case studies—from TCGA multi-omics exploration to vision-language retrieval on Food101—demonstrate iMML's capability and robustness under various modality missingness.

Major strengths
1. Incomplete multimodal data are ubiquitous in real-world applications, yet there is no well-established toolkit to handle missingness across modalities. iMML directly addresses this important challenge.
2. By unifying over 25 methods that cover methods including imputation, clustering, classification, and beyond. iMML offers a one-stop platform, and researchers no longer need to tackle disparate codebases with conflicting dependencies.
3. The authors have translated most methods to pure Python. Continuous integration and > 97% test coverage can particularly help tackle the multimodal missingness issue.
4. A unified API consistent with pandas/NumPy/scikit-learn, extensive tutorials, and visualization tools lower the barrier to adoption for both green-hand and experienced practitioners.
5. The six case studies cover a broad spectrum of problem domains and show that state-of-the-art methods can be applied "out of the box" to multimodal missingness.

Major weaknesses and suggestions for improvement
1. While the case studies are illustrative, the manuscript would be strengthened by a more systematic benchmark:
• While the 10x speedup (Table 2) is impressive, "similar behaviors" is too vague for a methodological benchmark. For stochastic methods, please provide a statistical comparison to prove that the Python implementation does not deviate statistically from the original Matlab/R code. For deterministic algorithms, quantify "exact match" (e.g., maximum absolute error between the output matrices).
• Scalability experiments (different training steps, dataset size, and network parameters) on larger datasets to characterize performance and limitations. For instance, how do memory usage and computation time scale for 1-2 key algorithms as the number of samples increases from 10^2 to 10^5, and modalities increase?

2. The current implementation primarily addresses numeric/tabular representations, but Section 4.1.1 mentions models like RAGPT, ViLT, and MUSE for images and text. Clarify how raw unstructured data (images, text) enters the iMML pipeline. A brief architectural flowchart in the supplementary material showing how raw inputs are processed would help to clarify this for deep learning practitioners.

3. The inclusion of 25+ algorithms is fantastic, but the manuscript lacks a brief taxonomy. Please briefly explain the criteria used to select these specific 25 algorithms. Do they represent the current State-of-the-Art (SOTA)? Are they the most highly cited?

4. It would be helpful to include a summary table of the exact hyperparameters used in each case study (batch sizes, learning rates, number of components, etc.).

5. Comparison to alternative toolkits. Table 1 mentions other libraries but does not benchmark them. A brief discussion of

trade-offs in terms of ease of use and extensibility would sharpen the positioning of iMML.

6. For Food101, only MCC is reported at the end of training. No confidence intervals, no repeated runs, no test on the significance of the MCC drop from complete to 70% incomplete.

7. For 2.2.1, the claim that CNA+Proteomics is "particularly informative" for survival is speculative ("we hypothesize") without any survival analysis or formal test. Adding a reference or experiment analysis will be great.

Minor comments
– Figure 1c: “Others” includes “survival analysis,” which could split the “survival analysis” from it, to make it more prominent, given the biomedical focus.
– Typos: “Expotential” → “Exponential” in Fig. 1 caption.
- The specific mathematical formulation of PID used here should be explicitly cited and briefly defined in the Methods, as PID has multiple definitions in information theory.
- Does the package support scenarios where a model is trained on complete multi-modal data but must perform inference on a sample with a missing modality? If so, this should be explicitly highlighted as it is a highly requested feature in clinical AI deployments.

(Remarks on code availability)

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The authors have addressed my comments in the previous version adequately. I thank them for their effort. I think the manuscript is now acceptable for publication. I believe iMML will be a useful tool for the research community working on multimodal learning.

(Remarks on code availability)

Point-by-point response

We sincerely thank the Editor and the two expert reviewers for their careful reading of our manuscript and for their constructive suggestions, which have helped us substantially improve the clarity, scope, and positioning of the work. We have now revised the manuscript accordingly and provide detailed responses below, together with the description of changes introduced in the revised version.

Reviewer #1

Summary

This manuscript presents iMML, a Python package for multi-modal learning with incomplete data, integrating more than 25 existing algorithms and providing unified interfaces for preprocessing, imputation, clustering, classification, and analysis. The software is well documented, maintained, and reproducible, and the provided tutorials demonstrate its usability across several domains. The work addresses an important practical gap in the field, namely the lack of standardized tools for incomplete multi-modal learning.

Strengths

1. Strong practical relevance. Handling incomplete multi-modal data is a common and challenging problem across many domains such as healthcare, bioinformatics, and social sciences. By providing a unified implementation of a wide range of existing methods, the proposed package significantly lowers the barrier for practitioners and researchers working in this area.

2. Comprehensive and well-integrated framework. The library integrates more than 25 algorithms covering multiple tasks and methodological families. The unified interface for preprocessing, simulation of missingness, model training, and evaluation makes the framework convenient for both experimentation and applied use.

3. High-quality software engineering. The repository is well structured, actively maintained, and easy to install and run. The documentation, tutorials, and example workflows are clear and helpful. The code appears robust and reproducible, making it a useful resource for the community.

4. Emphasis on reproducibility and usability. The inclusion of simulation tools, standardized workflows, and compatibility with common Python ecosystems (NumPy, PyTorch, scikit-learn) greatly enhances the usability of the package and supports reproducible research.

Response. We are very grateful to the reviewer for these positive comments and for appreciating the importance of standardized tools for multi-modal learning with incomplete data. We thank the reviewer for the thoughtful evaluation of our work and for the constructive comments that helped us better define the unique contributions of iMML.

Remarks on code availability

The repository is publicly available and well organized. The project structure is clear, with modular components for data preprocessing, missingness simulation, model implementation, and evaluation. The code follows a consistent design pattern and integrates well with common Python ecosystems such as NumPy, PyTorch, and scikit-learn. The package provides comprehensive documentation through an online documentation site, including an overview of the framework, installation instructions, API references, and multiple

usage tutorials. The README and documentation contain sufficient guidance for setting up the environment and running the main functionalities. The provided examples and notebooks help users understand typical workflows and reproduce the experiments presented in the manuscript. The installation process is straightforward, and the core functionalities can be executed without major issues. The implementation appears stable and actively maintained. The modular design also makes it relatively easy for users to extend the framework or integrate their own methods. Regarding reproducibility, the available scripts and examples allow users to reproduce the main experimental pipelines described in the paper at a functional level. While some experimental settings (e.g., dataset-specific preprocessing details or hyperparameter configurations) could be further specified for exact numerical reproduction, the current resources are sufficient to reproduce the methodology and overall results. Overall, the code is well documented, usable, and represents a valuable resource for the community.

Response 0. We thank the reviewer for these positive comments and for the careful evaluation of our software and documentation. We appreciate the recognition of the repository structure, documentation, usability, and reproducibility. In the revised manuscript, we have provided additional details on the hyperparameter configurations in the new Table S4. We note that most datasets were already preprocessed in their original sources. When additional preprocessing was required, the specific procedures have been described more clearly in the Methods section for each use case (pages 17-18). In addition, we have made all codes openly available to ensure that the results can be reproduced, specified under the Code availability section.

Suggestions for Improvement

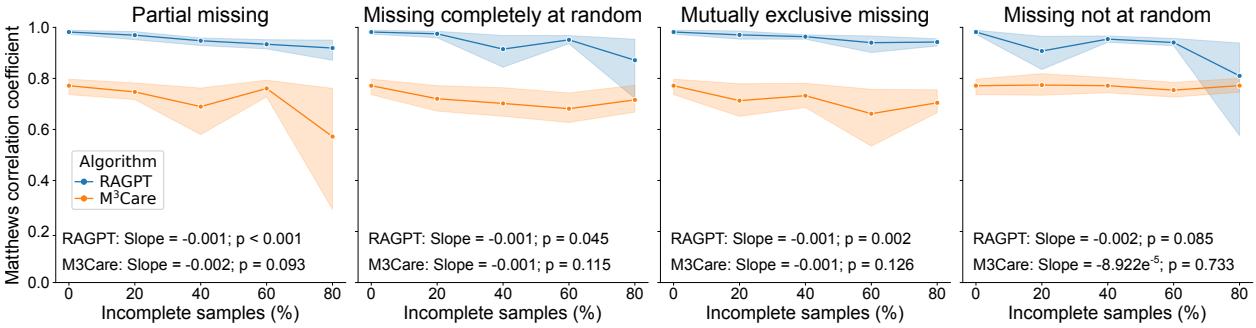
1. Evaluation is illustrative rather than systematic. The experimental case studies effectively demonstrate the usage of the package and illustrate its flexibility. However, the current evaluation appears primarily demonstrative rather than systematic. It would be helpful to clarify whether the presented results are intended as usage examples rather than comprehensive benchmarking. If possible, adding a more structured comparison (e.g., standardized datasets, consistent settings across multiple methods) would further strengthen the empirical support.

Response 1. We thank the reviewer for this well-received comment. The main contribution of this work is the iMML framework itself, and the experimental section was originally designed primarily to illustrate how the package can be used in practice across representative tasks and data types, as usage examples. However, we agree that the original version would benefit from stronger empirical evidence. We have therefore revised the manuscript to make the purpose of the experiments explicit and to strengthen the comparisons where possible. In particular, we have expanded the clustering and classification case studies by including additional methods, and standardized evaluation settings, as requested, which makes the results more informative and easier to interpret and compare across multiple methods. These results are added to Results sections 2.3.3 (page 10, lines 228-233) and 2.3.4 (page 10, lines 245-246, 251-253, and 257-258), and we have updated Methods section 4.3.2 (page 17, lines 526-532), Fig. 4 and Supplementary Fig. S5, and added a new Supplementary Fig. S4.

For classification, we compared the two vision-language classification algorithms currently available in iMML, namely RAGPT and M^{3}Care. We further introduced a new evaluation module in iMML that facilitates train/test splits, cross-validation, stratified cross-validation and related workflows in multi-modal datasets. Using this new module, we now report results with a stratified five-fold cross-validation with consistent hyperparameter settings under four missingness patterns and increasing incompleteness rates of 0%, 20%, 60%, and 80%. Because this comparison substantially increased computational requirements (approximately one week on an NVIDIA RTX 2080 Ti GPU), we used a binary subset of the Food101 dataset comprising the two most frequent classes (1,424 samples).

The new results show that RAGPT is highly robust to missing modalities and consistently achieves the best performance across all evaluated conditions (New Supplementary Fig. S4). M^{3}Care also demonstrates strong robustness to incompleteness, although its overall predictive performance was lower.

For clustering, we now evaluate five methods (IMSCAGL, IMSR, LFIMVC, PIMVC, SIMCADC) under four missingness patterns and compare them against a mean-imputation baseline in the same BBCSport dataset. These results show that methods designed for incomplete data consistently outperform prior imputation across the tested settings. More specifically, IMSR performs best across most tested conditions,



New Supplementary Figure S4: **Classification performance on an incomplete vision-language dataset**. Stratified five-fold cross-validation of RAGPT and M^{3}Care algorithms on the Food101 dataset across various missing rates (from 0% to 80%) and different data missingness patterns using Matthew's correlation coefficient. A linear model was fitted for each model and missingness pattern to estimate how the performance varies when using incomplete data. Results are averaged across folds, with 95% confidence intervals indicated.

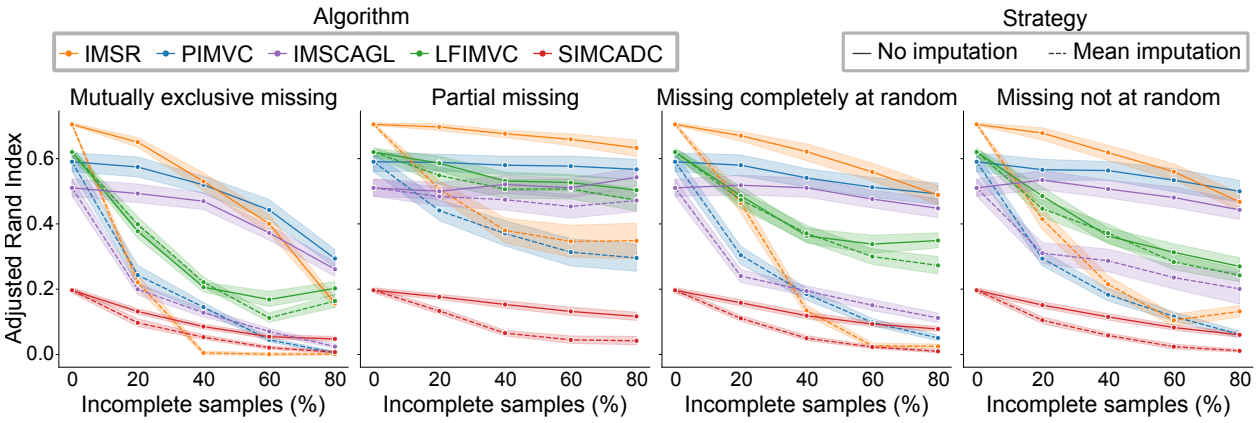
whereas PIMVC is a very competitive option at high missingness rates (Updated Supplementary Fig. S5).

We hope these additional comparisons provide a more complete and transparent illustration of the methods' behavior under incomplete multi-modal data. We have further clarified in the manuscript (page 13, lines 354-361) that a large-scale benchmark across datasets, missingness mechanisms, robustness criteria, and computational cost is an important future direction.

2. Missingness assumptions and robustness claims could be clarified The manuscript highlights the capability of the framework to handle different missingness scenarios. However, many of the integrated methods were originally developed under specific assumptions (e.g., MCAR or MAR). It would be beneficial to clarify:

- **2.1**. Under what assumptions the included methods are expected to perform reliably.

  Response 2.1. We thank the reviewer for raising this essential point. In the revised manuscript, we clarify that the methods included in iMML are selected because they can operate on incomplete multi-modal data without requiring specific missingness pattern (Supplementary Information S1.2). Therefore, all the methods should be able to handle any missingness pattern as long as the missingness is modality-wise. Among the collected methods, only DFMF, MOFA and jNMF are able to handle both single-value and modality-wise missing data. Methods for single-value missing data have already been



Updated Supplementary Figure S5: **Clustering performance on incomplete multi-modal text data**. The clustering solutions by IMSCAGL, IMSR, LFIMVC, PIMVC and SIMCADC on the BBCSport dataset, as well as their respective baselines (where missing values were replaced with the feature-wise mean), are compared to ground truth topics across various missing rates (from 0% to 80%) under different data missing patterns using adjusted rand index. The average of 50 repetitions with 95% confidence intervals are shown.

New Supplementary Table S3: **Conditions tested during algorithm development in the original publications**. None of the method included in iMML is based on a specific mathematical assumption for the missingness pattern, and therefore each can operate on any missingness pattern; this table shows which missingness patterns were used by the authors to validate the algorithms. Algorithms or conditions that were not fully described or could not be inferred from the paper were not included in this table.

Algorithm	Missingness patterns				Missing data		Maximum Rate (%)
	MCAR	MEM	MNAR	PM	Simulated	Real	
DAIMC			✓		✓		50
EEIMVC	✓				✓		90
IMSCAGL	✓				✓		90
IMSR	✓				✓		50
IntegrAO	✓			✓	✓	✓	90
LFIMVC	✓				✓		90
M^{3}Care				✓	✓	✓	60
MKKMIK	✓				✓		90
MRGCN				✓	✓		70
MOFA	✓			✓	✓	✓	90
MONET	✓		✓		✓		40
MUSE	✓			✓	✓	✓	90
NEMO				✓	✓		80
OMVC			✓		✓		40
OPIMC			✓		✓		50
OSLFIMVC	✓				✓		50
PID				✓	✓		50
PIMVC			✓	✓	✓		70
RAGPT		✓		✓	✓		90
SIMCADC			✓		✓		50
SUMO				✓	✓	✓	90

Abbreviations: MCAR, missing completely at random; MEM, mutually exclusive missing; MNAR, missing not at random; PM, partial missing.

studied extensively, which is why we focused our work on incomplete multi-modal data. However, it is true that the empirical validation settings used in the original works differ across methods, reflecting the view of the original authors about the missingness scenarios they were thinking when developing their methods. To make this more transparent, we have now added a supplementary table summarizing the missingness patterns and incompleteness levels used in the original papers when each method was developed and evaluated (New Supplementary Table S3). This table helps the reader distinguish between the broad applicability of the iMML package and the more specific conditions under which each algorithm was originally developed and validated.

- **2.2**. Whether robustness to different missingness mechanisms has been systematically evaluated.

  Response 2.2. Our goal was to present a unified and extensible framework with a Python package, and to show how it can be used consistently across representative learning scenarios. We have therefore included multiple illustrative examples showing how missingness pattern and missingness rate affect the methods (please see Fig. 4 in the manuscript), but we did not aim to provide a full benchmark of robustness across all missingness mechanisms. As described above, the original publications have tested the methods under various missingness mechanisms, which provides a good stating point for the end-users of the iMML package. Given its scope, such a systematic benchmark would constitute a separate study, because it would require careful comparison across multiple datasets, missingness mechanisms, tasks, and computational settings. However, we agree that this would be extremely useful for the community, and we have therefore added this point to the revised Discussion as a future direction (page 13, lines 354-361).

- **2.3**. The extent to which the framework itself, versus the individual methods, provides robustness.

  Response 2.3. We thank the reviewer for this important clarification request. Our framework provides

improved robustness to missing data, compared with other open-source frameworks featuring multiple methods, which, as described in Table 1 (in the manuscript), do not currently support incomplete multi-modal data.

Regarding the underlying algorithms, iMML does not alter their mathematical logic, and their performance is determined by the original methods. Therefore, the original implementations of the methods and our package are expected to produce the same algorithmic performance when run under the same conditions. However, iMML package offers several important advantages in terms of technical robustness and reproducibility relative to the original implementations:

- Most original implementations were developed for a specific dataset and often restricted to a fixed number of modalities or data formats. We generalized these implementations to support arbitrary numbers of modalities and a broader range of data structures, substantially increasing their practical applicability.
- We noticed that many of the original methods lacked a means to control for stochasticity. To fix this, and following scikit-learn approach, we introduced a random_state parameter in all methods, so that repeated runs with the same experiment and random_state return exactly the same result. This is crucial for reproducibility, because otherwise results cannot be reliably replicated.
- Except for MOFA or NEMO, the shared code in the original implementations was intended to reproduce the results of the corresponding articles, rather than to provide a tool that others could easily use. As software ecosystems evolve, such code often becomes unusable due to dependency conflicts. We have therefore made a comprehensive effort to minimize package dependencies, ensuring only the requirement of widely used and well-maintained libraries (Pandas, Numpy, scikit-learn, Lightning AI).
- In addition, iMML uses version control, tags and released versions through GitHub and the Python Package Index (PyPI), which means that each use of the methods is tied to a specific set of requirements, independently of when it is executed.
- For deep learning models, iMML integrates with Lightning AI, a powerful deep learning framework that provides full control over the machine learning workflow and makes it easier to handle advanced features such as GPU parallelization, model quantization, or automatic learning-rate search.

Additionally, one of the main advantages of the iMML package is that all methods share the same interface, which reduces the technical knowledge required to use different algorithms and lowers code complexity, making it easier to benchmark and compare multiple methods across various datasets, missingness patterns and rates.

We further note that the package-level contribution is not to alter the mathematical behavior of the underlying algorithms, but to provide a unified, reproducible, and user-friendly environment in which they can be fairly compared and applied. Accordingly, iMML improves technical robustness through standardized interfaces, reproducible seeds, consistent data handling, and shared evaluation utilities, while the algorithmic robustness remains determined by each method's own design.

We have added this clarification to the Methods section 4.1.3 (page 16, lines 488-492) and Discussion (pages 12-13, lines 332-353).

Reviewer #2

Summary

This manuscript proposes iMML, an open-source Python package designed to fill the gap in the multi-modal learning community: handling incomplete data in a unified, user-friendly framework. iMML integrates over 25 published algorithms for tasks such as imputation, clustering, classification, retrieval, feature extraction/selection, and visual exploration. The authors have implemented most methods in Python, wrapped the remaining Matlab/R code for seamless use, and provided a consistent scikit-learn–style API, extensive

documentation, and high coverage unit testing across platforms. Six illustrative case studies—from TCGA multi-omics exploration to vision-language retrieval on Food101—demonstrate iMML's capability and robustness under various modality missingness.

Strengths

1. Incomplete multimodal data are ubiquitous in real-world applications, yet there is no well-established toolkit to handle missingness across modalities. iMML directly addresses this important challenge.
2. By unifying over 25 methods that cover methods including imputation, clustering, classification, and beyond. iMML offers a one-stop platform, and researchers no longer need to tackle disparate codebases with conflicting dependencies.
3. The authors have translated most methods to pure Python. Continuous integration and $> 97\%$ test coverage can particularly help tackle the multimodal missingness issue.
4. A unified API consistent with pandas/NumPy/scikit-learn, extensive tutorials, and visualization tools lower the barrier to adoption for both green-hand and experienced practitioners.
5. The six case studies cover a broad spectrum of problem domains and show that state-of-the-art methods can be applied "out of the box" to multimodal missingness.

Response. We sincerely thank the reviewer for appreciating the strengths and novelty of our work and for the thoughtful, constructive and detailed comments. We greatly appreciate the opportunity to clarify the scope of the framework and to markedly strengthen the revised manuscript accordingly.

Major weaknesses and suggestions for improvement

1. While the case studies are illustrative, the manuscript would be strengthened by a more systematic benchmark:

- **1.1**. While the 10x speedup (Table 2) is impressive, "similar behaviors" is too vague for a methodological benchmark. For stochastic methods, please provide a statistical comparison to prove that the Python implementation does not deviate statistically from the original Matlab/R code. For deterministic algorithms, quantify "exact match" (e.g., maximum absolute error between the output matrices).

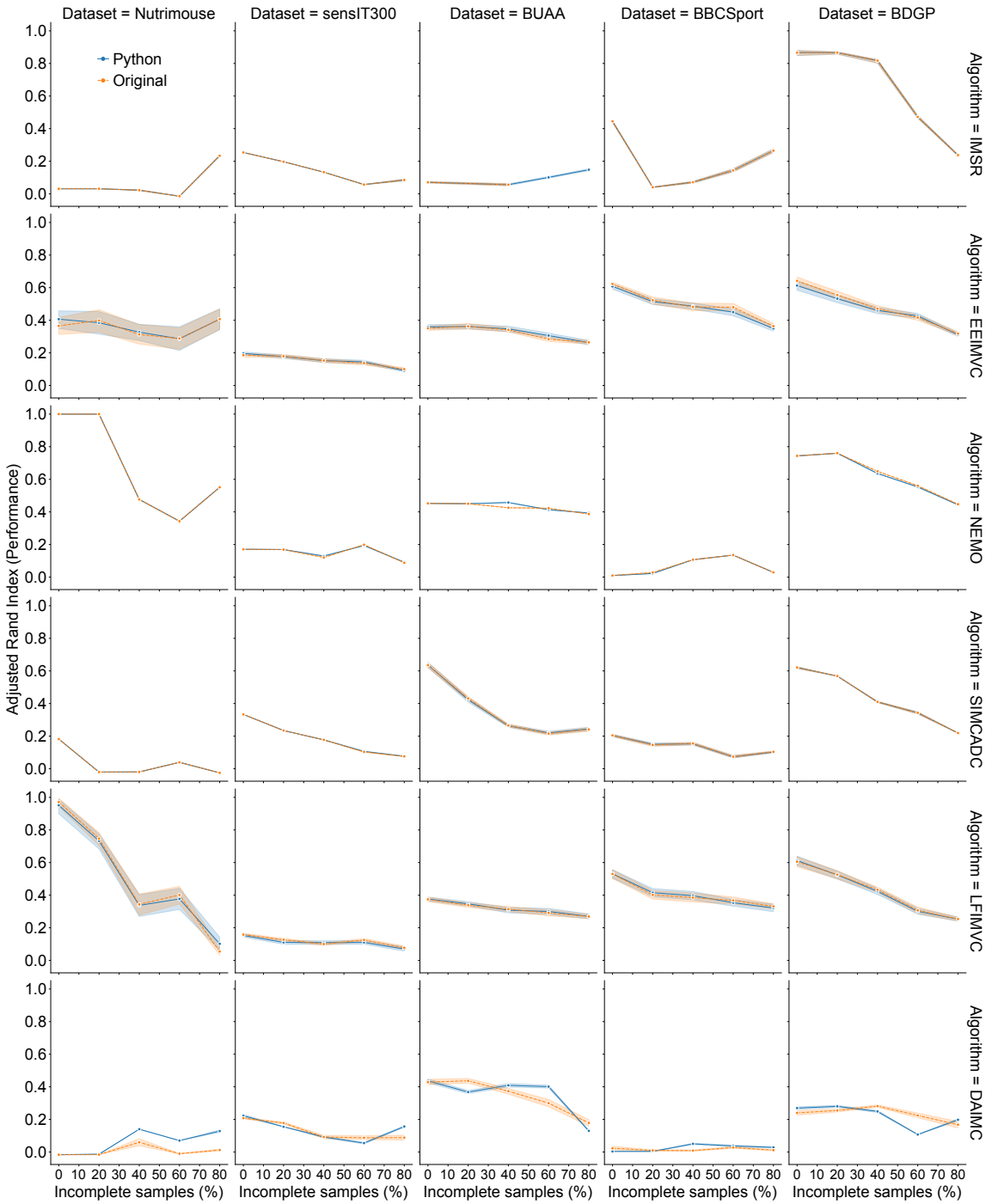
  Response 1.1. We thank the reviewer for this important technical request. We have now carefully re-evaluated the new Python implementations available in iMML against the original Matlab/R versions to evaluate both performance and stability under identical preprocessing and experimental settings. To address this comment, we have revised the manuscript to make the evaluations more explicit and quantitative, by reporting paired statistical comparisons between the original and translated implementations using the Nemenyi-Wilcoxon-Wilcox test.

  The observed p-values are consistently non-significant ($p = 1$ for all methods), indicating that the new Python implementations do not deviate statistically from the original Matlab/R codes (New Supplementary Fig. S1.

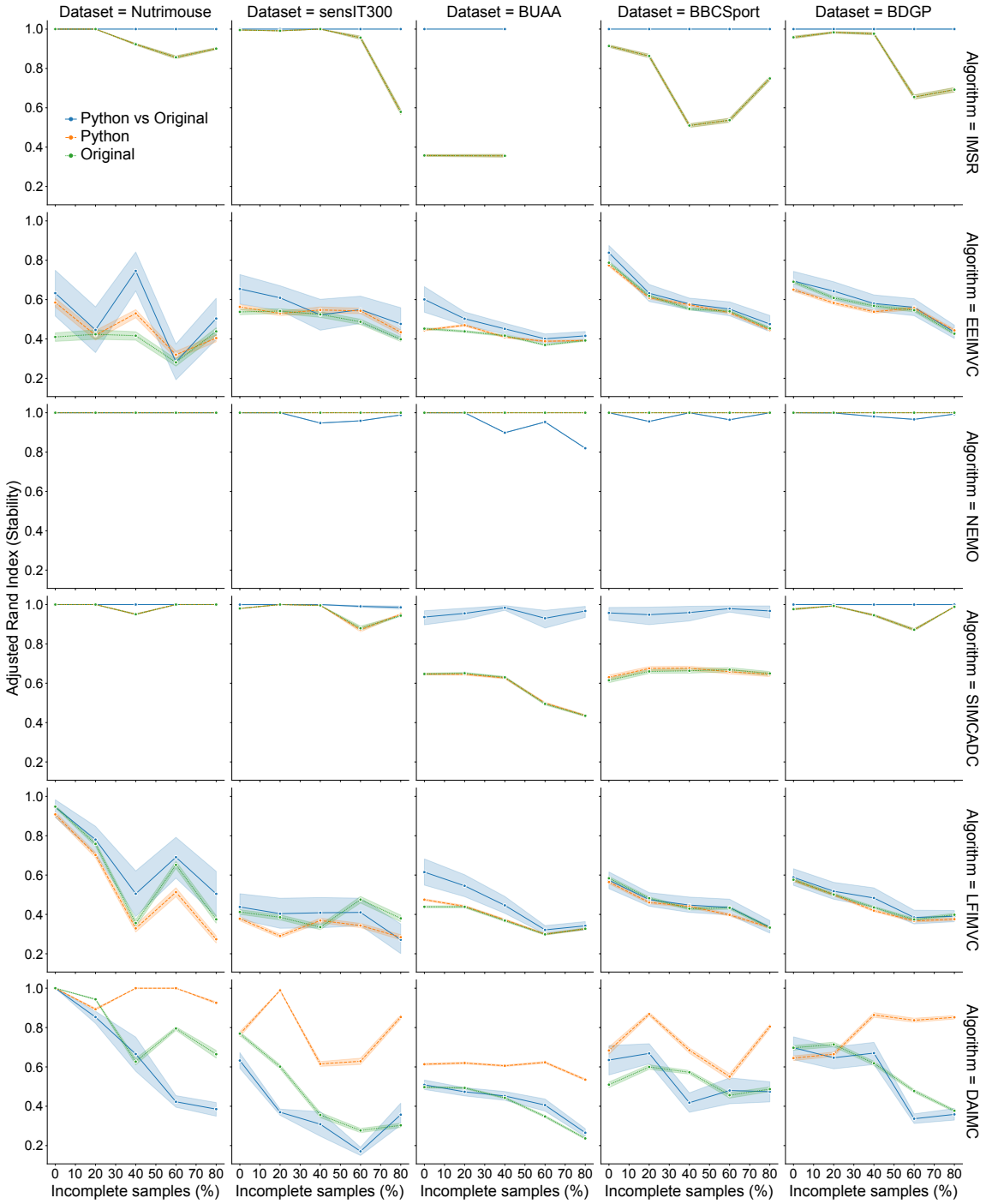
  For DAIMC, the results are somewhat more variable, but the two implementations remain highly equivalent, and the performance is clearly preserved. This slight variability was expected because the algorithm is highly stochastic, as even comparisons across repetitions within the same programming language do not yield identical results (New Supplementary Fig. S2).

  In terms of quantifying the error in the matrices for deterministic methods, NEMO was the only deterministic clustering algorithm (New Supplementary Fig. S2). The difference between the new Python implementation and the original implementation was due only to floating-point precision in intermediate steps across languages, with an average $RMSE = 1.72e-16$.

  We also re-evaluated the computational performances and found that the Python implementations remain overall substantially faster, with speed-ups ranging from 1–10×. Importantly, this speed improvement is not achieved at the expense of memory usage (Updated Table 2).



New Supplementary Figure S1: **Performance comparison between the original and the new Python implementations**. Six clustering algorithms (rows) were translated to Python and evaluated on five multi-modal datasets (columns). Clustering solutions were compared against ground truth using adjusted Rand index. Larger values indicate better performance. For all algorithms, the results of the two implementations are nearly identical, with all $p = 1$ using Nemenyi-Wilcoxon-Wilcox test, indicating no evidence against the null hypothesis that the implementations produce the same distributions. IMSR failed to converge in the original implementation on the BUAA dataset when the proportion of incomplete samples exceeds 60%. Results are averaged over 50 repetitions, with 95% confidence intervals indicated.



New Supplementary Figure S2: **Stability comparison between the original and the new Python implementations**. Stability was measured by pairwise comparison of clustering solutions across repetitions using adjusted Rand index. Larger values indicate greater stability. We assessed the stability of the clustering solutions by repeating the algorithms 50 times within the same programming language, and also comparing the clustering solutions between the two programming languages. Stability for IMSR on the BUAA dataset was not evaluated when the proportion of incomplete samples exceeded 60%, since the original implementation did not converge under those conditions. For NEMO (a deterministic method), the difference was due to floating-point precision in intermediate steps across languages, with an average $RMSE = 1.72e-16$. Results are averaged across repetitions, with 95% confidence intervals indicated.

Updated Table 2: **Computing performance comparison between the original and the new Python implementations**. For each algorithm, the average speed-up and memory gain over five datasets and 50 repetitions are reported, with minimum and maximum values indicated in parentheses. Values larger than 1 indicate improved performance of the Python implementation (e.g., a value of 2 corresponds to a two-fold speed-up or a two-fold reduction in memory usage). The Python implementations show clear speed improvements and lower memory consumption in most cases.

Algorithm	Speed-up factor (×)	Memory reduction factor (×)
LFIMVC	9.33 (3.08, 16.55)	1.82 (1.09, 3.20)
EEIMVC	8.45 (1.78,16.23)	1.54 (1.07,2.35)
NEMO	5.61 (2.45,10.81)	2.02 (1.01,4.71)
SIMCADC	5.47 (2.19,10.64)	1.63 (1.10,2.26)
IMSR	3.69 (1.11,13.86)	1.21 (0.44,2.23)
DAIMC	0.97 (0.24,2.09)	1.50 (1.03,2.19)

These results are now discussed in the revised Results section 2.2 (page 6, lines 138-145), and both Table 2-S1 and Methods section 4.2 (page 16, lines 499-503) have been updated accordingly. In addition, Supplementary Fig. S3-S9 have been removed so that these validations are now presented more compactly in the new Supplementary Fig. S1 and S2.

- **1.2**. Scalability experiments (different training steps, dataset size, and network parameters) on larger datasets to characterize performance and limitations. For instance, how do memory usage and computation time scale for 1-2 key algorithms as the number of samples increases from 10^2 to 10^5, and modalities increase?

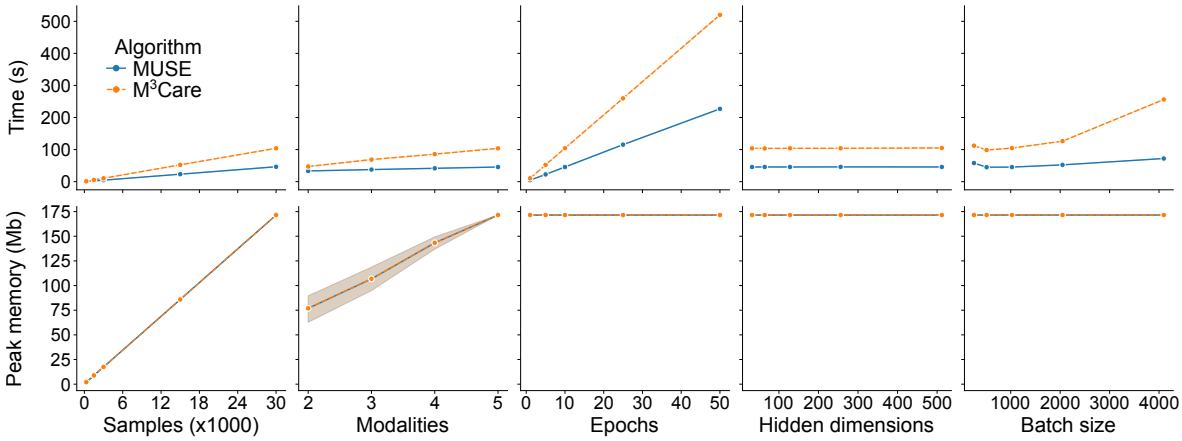
  Response 1.2. We thank the reviewer for suggesting this important addition. We have now included a scalability analysis for two state-of-the-art classification algorithms, MUSE and M^{3}Care, using the NUSWIDE dataset. NUSWIDE is a widely-used real-world web image dataset with 5 modalities and 30,000 samples, in which each image is represented by five types of low-level features, i.e., color histogram, color correlogram, edge direction histogram, wavelet texture, and block-wise color moments [1]. The new experiment evaluates runtime and memory usage across 10 repeated runs and shows how the performance changes with increasing sample size (300, 1500, 3000, 15000 and 30000), batch size (256, 512, 1024, 2048, 4096), hidden dimensions (32, 64, 128, 256, 512), number of epochs (1, 5, 10, 25, 50), and modality complexity (from 2 to 5 modalities). This experiment evaluates how these algorithms scale when varying the data, including the number of samples and modalities, the training parameters, including the number of epochs (training steps) and batch size, and the network parameters, including the hidden dimensionality of the model.

  The results indicate that the number of samples has the strongest impact on peak memory for both algorithms, followed by the number of modalities (new Supplementary Fig. S3). For runtime, the number of epochs is the most important factor. This evaluation helps clarify the practical strengths and limitations of the framework in larger-scale settings.

  This new analysis has been described in the Results section 2.2 (page 6, lines 149-155) and Methods section 4.2 (page 16, lines 504-512), and the results added to the new Supplementary Fig. S3.

2. The current implementation primarily addresses numeric/tabular representations, but Section 4.1.1 mentions models like RAGPT, ViLT, and MUSE for images and text. Clarify how raw unstructured data (images, text) enters the iMML pipeline. A brief architectural flowchart in the supplementary material showing how raw inputs are processed would help to clarify this for deep learning practitioners.

Response 2. We thank the reviewer for this helpful clarification request. We have now revised the manuscript to explain that iMML expects each modality to be represented as a matrix-like object, such as a NumPy array, Pandas dataframe, or PyTorch tensor (page 16, lines 479-481). For unstructured modalities, such as text and images, the input is also standardized: text is stored as a one-column representation containing the raw text entries, while image inputs are stored as paths so that loading can occur efficiently within each batch and avoid memory overload. Missing values are represented as NaN, which allows the same pipeline to handle structured and unstructured modalities in a unified way. To make this workflow



New Supplementary Figure S3: **Scalability comparison**. MUSE and M^{3}Care were evaluated for computing time (top) and peak memory (bottom), while varying the number of samples, modalities, epochs, hidden dimensions and batch size on the large-scale NUSWIDE dataset. Runtime is strongly dependend on the number of epochs, whereas peak memory is most strongly affected by the number of samples, followed by the number of modalities. Only a single variable was adjusted at a time, while the others remained at their default values (samples = 30000, modalities = 5, epochs = 5, hidden dimensions = 128, batch size = 1024). Results are averaged over 10 repetitions, with 95% confidence intervals indicated.

easier to access, we have added a supplementary schematic that illustrates the input format and processing flow from raw modality-specific data to the method interface (New Supplementary Fig. S10).

3. The inclusion of 25+ algorithms is fantastic, but the manuscript lacks a brief taxonomy. Please briefly explain the criteria used to select these specific 25 algorithms. Do they represent the current State-of-the-Art (SOTA)? Are they the most highly cited?

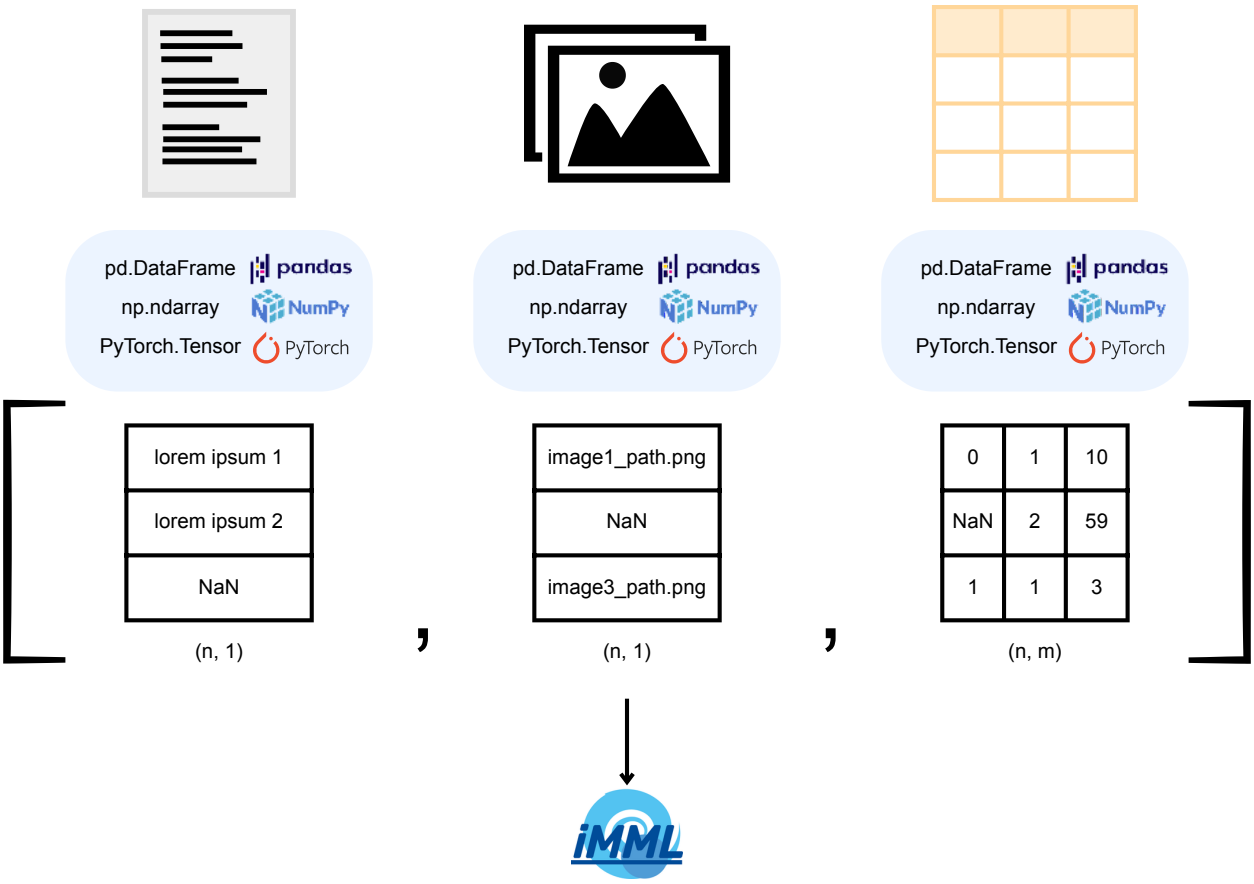
Response 3. iMML includes established, emerging and state-of-the-art algorithms. To address this question more clearly, we have revised the manuscript to explain that the included methods were selected according to a set of practical and methodological criteria (Supplementary Information S1.2). First, each method must be able to operate without requiring fully observed samples during either training or testing, and without assuming a specific missingness pattern. Second, a method must support an arbitrary number of modalities, rather than being tied to a very narrow multi-modal setting. Third, a method should not be restricted to a specific data type (such as biological pathways, hyperspectral images, histopathological images), so that the package remains relevant across multiple application domains. Fourth, we prioritized methods that are highly cited (Table 1). Fifth, the code should be open and sufficiently documented.

Methods that did not meet these criteria were set aside for potential future inclusion. However, to maximize the practical impact of the package, we also incorporated a subset of methods specifically tailored to vision-language tasks, which represent a prominent use case in the field. For this particular use case, we included RAGPT (state-of-the-art for image-language classification), as well as MUSE and M^{3}Care (which support multiple modality types, such as images, text, time-series and tabular data).

In summary, the methods that have been included are general enough to handle any number of modalities, and all of them can work with complete or incomplete multi-modal datasets without requiring specific missingness pattern. Thus, the package is designed to provide a broad and useful collection of established, emerging and state-of-the-art algorithms.

4. It would be helpful to include a summary table of the exact hyperparameters used in each case study (batch sizes, learning rates, number of components, etc.).

Response 4. We thank the reviewer for this practical suggestion. We have prepared a summary table of the key hyperparameters used in each case study, including batch size, number of components, and other method-specific settings where applicable (New Supplementary Table S4). This addition improves reproducibility and makes it easier for readers to replicate the reported experiments.



New Supplementary Figure S10: **Input format and processing flow in iMML package**. The package accepts a list of matrix-like objects, such as a NumPy array, Pandas dataframe, or PyTorch tensor for text, images and tabular data. Text is stored as a one-column representation containing the raw text entries, while image inputs are stored as paths so that loading can occur efficiently within each batch and avoid memory overload. Missing values are represented as NaN, which allows the same pipeline to handle structured and unstructured modalities in a unified way. This constitutes the input format for methods implemented in iMML, so the users do not need to add any preprocessing steps, as each algorithm applies its own transformation. Typically, images are loaded using the PIL library, converted to a PyTorch tensor, and then encoded using a convolutional neural network or visual transformer; texts are tokenized and encoded with a language model such as BERT [2]. n is the number of samples, and m is the number of features in a table.

5. Comparison to alternative toolkits. Table 1 mentions other libraries but does not benchmark them. A brief discussion of trade-offs in terms of ease of use and extensibility would sharpen the positioning of iMML.

Response 5. We thank the reviewer for this useful request. We have now added a new Results section 2.1 to compare our framework with the existing libraries (pages 5-6). Comparing iMML with the other popular open-source packages, as summarized in Table 1 (in the manuscript), we note that none of the existing libraries addresses the challenge of multi-modal learning with incomplete data:

- **Imputation and amputation**. HyperImpute, Scikit-learn and mdatagen offer a variety of imputation methods. The popular R mice package is widely used for both imputation and amputation. Another popular library for data amputation is pyampute. While these libraries are useful and the common choices for imputing missing data, they are limited to uni-modal data and do not support multi-modal contexts.
- **Multi-modal learning**. mvlearn offers a range of algorithms for different tasks, following a scikit-learn style interface. However, we found it is no longer maintained (last commit in 2022). In addition,

Table 1: **Citation counts of each method included in the iMML package**. Note that the more recent methods are likely to have attracted fewer citations. Citations are based on Google Scholar, accessed on May 2026.

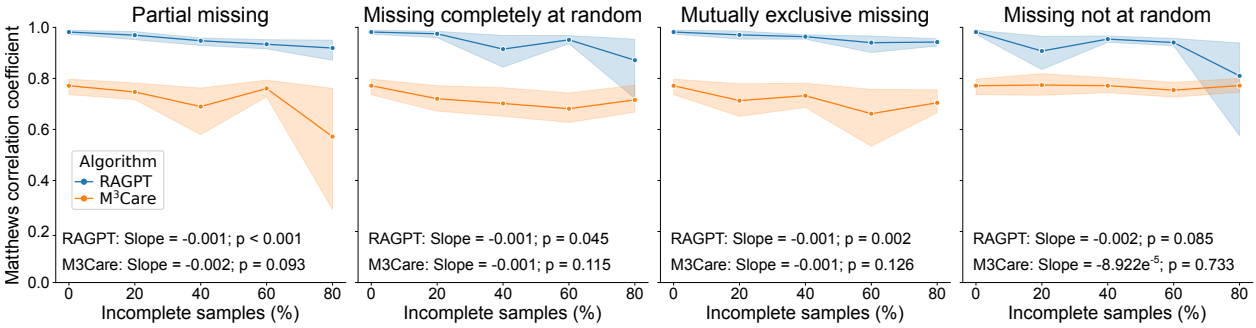
Algorithm	Citations	Year	Programming language
MOFA	1593	2018	Python
MKKMIK	538	2020	Matlab
LFIMVC	420	2019	Matlab
IMSCAGL	414	2020	Matlab
DAIMC	391	2018	Matlab
EEIMVC	380	2021	Matlab
JNMF	343	2016	R
NEMO	312	2019	R
DFMF	263	2015	Python
OPIMC	188	2019	Matlab
M^{3}Care	179	2022	Python
OMVC	168	2016	Matlab
PIMVC	133	2023	Matlab
IMSR	122	2021	Matlab
PID	102	2023	Python
MUSE	63	2024	Python
OSLFIMVC	58	2021	Matlab
MONET	31	2020	Python
MRGCN	28	2023	Python
RAGPT	24	2025	Python
SIMCADC	24	2023	Matlab
IntegrAO	21	2025	Python
SUMO	16	2022	Python

New Supplementary Table S4: **Hyperparameters of the methods used in the experiments**. For other hyperparameters not indicated in this table, default parameters in the library were used (iMML v0.3.1). When a list is indicated, benchmarking with all the values was performed and results provided.

Use case	Algorithm	Hyperparameters
Classification	RAGPT	Batch size=32, n_neighbors=20, epochs=10
Classification	M^{3}Care	Batch size=32, epochs=10
Clustering	IMSCAGL	Clusters=5 (#classes)
Clustering	IMSR	Clusters=5 (#classes)
Clustering	LFIMVC	Clusters=5 (#classes)
Clustering	PIMVC	Clusters=5 (#classes)
Clustering	SIMCADC	Clusters=5 (#classes)
Imputation	MOFA	Components=[1,2,4,8,16]
Dimensionality reduction	jNMF	Components=[1,2,4,8,16]
Feature selection	jNMF	Components=[1,2,4,8,16]

scikit-multimodallearn follow the same API design, although it is limited to classification tasks only. However, mvlearn and scikit-multimodallearn only support fully observed multi-tabular data. MultiZoo/MultiBench is a repository (and therefore not actual package published in the Package Index, PyPI), which includes a large diversity of multi-modal datasets and fusion paradigms, optimization objectives, and training approaches. TorchMultimodal is another interesting library, although it includes only deep learning algorithms. The main problems we found was limited documentation, the most recent supported Python version is 3.9, and it is not maintained. None of the packages addresses learning with incomplete data.

- **Multi-modal learning with incomplete data**. The only notable repository in this space was developed in Matlab, where the authors tried to provide a framework to unify incomplete multi-view clustering [3]. The implementation in Matlab limits the accessibility and ease of use of the framework, which is why we translated the clustering algorithms available in this repository. Additionally, its scope is restricted to unsupervised clustering.



New Supplementary Figure S4: **Classification performance on an incomplete vision-language dataset**. Five-fold stratified cross-validation of RAGPT and M^{3}Care algorithms on the Food101 dataset across various missing rates (from 0% to 80%) and different data missingness patterns using Matthew's correlation coefficient. A linear model was fitted for each model and missingness pattern to estimate how the performance varies when using incomplete data. Results are averaged across folds, with 95% confidence intervals indicated.

In summary, none of the existing libraries addresses the challenges posed by incomplete multi-modal datasets. This is where iMML package fills a critical gap by providing researchers and the machine learning community with a tools-set and user-friendly environment to work effectively with incomplete multi-modal datasets.

6. For Food101, only MCC is reported at the end of training. No confidence intervals, no repeated runs, no test on the significance of the MCC drop from complete to 70% incomplete.

Response 6. We thank the reviewer for raising this important point. To improve our analysis on the Food101 classification task, we now compare the two vision-language classification algorithms currently available in iMML, namely RAGPT and M^{3}Care (Updated Supplementary Fig. S4). We further introduced a new evaluation module in iMML that facilitates train/test splits, cross-validation, stratified cross-validation and related workflows in multi-modal datasets. Using this new module, we now report results with a stratified five-fold cross-validation under the four missingness patterns and increasing incompleteness rates of 0%, 20%, 60%, and 80%. Because this analysis substantially increased computational requirements (approximately one week on an NVIDIA RTX 2080 Ti GPU), we used a binary subset of the Food101 dataset comprising the two most frequent classes (1,424 samples).

The new results show that RAGPT is highly robust to missing modalities and consistently achieves the best performance across all evaluated conditions. M^{3}Care also demonstrates strong robustness to incompleteness, although its overall predictive performance is lower. The performance drop, and p-values of such drop, are also reported (New Supplementary Fig. S4).

These results are added to Results sections 2.3.3 (page 10, lines 228-233), and we have updated Methods section 4.3.2 (page 17, lines 526-532), Fig. 4, and added the new Supplementary Fig. S4.

7. For 2.2.1, the claim that CNA+Proteomics is "particularly informative" for survival is speculative ("we hypothesize") without any survival analysis or formal test. Adding a reference or experiment analysis will be great.

Response 7. To address this comment, we have improved our analysis by making a more robust estimation of the multi-modal statistics:

1. Survival data was binarized at 18 months (selected between the median, 16, and the mean, 21). Censored patients alive earlier than 18 months were excluded.

2. The method was applied 25 times with different initialization seeds to improve robustness, and the results were averaged.

3. The best modality combination was selected based on total information (redundancy + uniqueness + synergy), i.e., the combination providing the most information for survival prediction.

Our new analysis suggest that the combination of gene expression and proteomics is particularly informative (new Supplementary Table S2). To the best of our knowledge, no systematic benchmarking of the best combination of data modalities for survival prediction using our four omics data types (proteomics, mutations, gene expression and copy number alterations) in pancreatic cancer exist to validate our approach. However, multiple proteogenomic studies in pancreatic cancer consistently show that proteomic data provides non-redundant and clinically relevant information beyond transcriptomics (median gene-wise correlation of 0.32 [4]), and that integrating RNA and protein layers improves survival association [5, 6].

We have revised the text to avoid overstatement and to better support the claim with existing literature (Results section 2.3.1 on pages 8-9, lines 185-194). Fig. 4 and the Methods section 4.3.1 have been updating accordingly (page 17, lines 516-523). The new Supplementary Table S2 has been also included.

New Supplementary Table S2: **Multi-modal statistics of survival prediction in pancreatic cancer**. The modality combination for predicting survival that provides most information is proteins and gene expression. Information refers to amount of information that the modality combination provides about survival prediction [7]. The table indicates the average across 25 repetitions.

Modality (1)	Modality (2)	Information	Redundancy (%)	Uniqueness (1) (%)	Uniqueness (2) (%)	Synergy (%)
Proteins	Gene expression	0.42	0.12	0.10	0.02	0.76
CNA	Proteins	0.40	0.17	0.05	0.06	0.72
CNA	Gene expression	0.36	0.17	0.08	0.00	0.74
CNA	Mutations	0.18	0.18	0.31	0.00	0.51
Mutations	Gene expression	0.17	0.18	0.02	0.17	0.63
Proteins	Mutations	0.17	0.18	0.36	0.02	0.44

Minor comments

- **8**. Figure 1c: "Others" includes "survival analysis," which could split the "survival analysis" from it, to make it more prominent, given the biomedical focus.

  Response 8. In the original visualization, we excluded those tasks with less than 5 papers. We have now broken down the category "Others" in the caption (page 2): regression (4), data alignment (3), generation (3), survival (2), tracking (2), explainability (1), question-answering (1).

- **9**. Typos: "Expotential" → "Exponential" in Fig. 1 caption.

  Response 9. We have fixed this typo, thank you for pointing it out (page 2).

- **10**. The specific mathematical formulation of PID used here should be explicitly cited and briefly defined in the Methods, as PID has multiple definitions in information theory.

  Response 10. We specifically refer to the mathematical formulation given by Liang *et al*[7]. We have now updated the Methods section 4.3.1 to be more explicit (page 17, line 518).

- **11**. Does the package support scenarios where a model is trained on complete multi-modal data but must perform inference on a sample with a missing modality? If so, this should be explicitly highlighted as it is a highly requested feature in clinical AI deployments.

  Response 11. Yes, this scenario is supported by the methods included in iMML. As noted in Response 3, one of the explicit criteria we used to select the methods was that they should be able to handle missing data in both training and test sets. Consequently, models implemented in iMML can be trained on complete multi-modal datasets and subsequently applied to samples with missing modalities. At the same time, we note that iMML does not automatically extend arbitrary external algorithms to handle missing modalities if they are not inherently designed for this purpose. However, iMML is a continuously evolving tool, and if future work proposes such an approach, we will add it to the framework. We have clarified this in the revised manuscript (page 15, lines 468-471).

References

[1] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[3] Jie Wen, Zheng Zhang, Lunke Fei, Bob Zhang, Yong Xu, Zhao Zhang, and Jinxing Li. A survey on incomplete multiview clustering. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(2):1136–1149, 2023.

[4] Yexin Tong, Mingjun Sun, Lingli Chen, Yunzhi Wang, Yan Li, Lingling Li, Xuan Zhang, Yumeng Cai, Jingbo Qie, Yanrui Pang, et al. Proteogenomic insights into the biology and treatment of pancreatic ductal adenocarcinoma. *Journal of Hematology & Oncology*, 15(1):168, 2022.

[5] Do Young Hyeon, Dowoon Nam, Youngmin Han, Duk Ki Kim, Gibeom Kim, Daeun Kim, Jingi Bae, Seunghoon Back, Dong-Gi Mun, Inamul Hasan Madar, et al. Proteogenomic landscape of human pancreatic ductal adenocarcinoma in an asian population reveals tumor cell-enriched and immune-rich subtypes. *Nature cancer*, 4(2):290–307, 2023.

[6] Benjamin Miao, Tung-Shing Mamie Lih, Yingwei Hu, and Hui Zhang. Signatures of pancreatic ductal adenocarcinoma uncovered by integrative multi-omics analysis. *Cancers*, 18(4):687, 2026.

[7] Paul Pu Liang, Yun Cheng, Xiang Fan, Chun Kai Ling, Suzanne Nie, Richard Chen, Zihao Deng, Nicholas Allen, Randy Auerbach, Faisal Mahmood, et al. Quantifying & modeling multimodal interactions: An information decomposition framework. *Advances in Neural Information Processing Systems*, 36:27351–27393, 2023.

We sincerely thank the Editor and the two expert reviewers for their careful reading of the revised manuscript and response.

Reviewer #1

Remarks to the Author

The authors have addressed my comments in the previous version adequately. I thank them for their effort. I think the manuscript is now acceptable for publication. I believe iMML will be a useful tool for the research community working on multimodal learning.

We greatly appreciate the reviewer's recognition of the value and potential impact of iMML for the research community working on multimodal learning.