dGanX: a Python Library to Generate MA-XRF Raw Data Out of RGB Images via a Distributed Computing Approach

Alessandro Bombini[1(✉)], Serena Barone[1,2], Marta Magalini[3], Fernando García-Avello Bofías[1], Daniele Spiga[4], Tommaso Tedeschi[4], and Chiara Ruberto[1,2]

[1] Istituto Nazionale di Fisica Nucleare, Sezione di Firenze, Sesto Fiorentino (FI), Italy
alessandro.bombini@fi.infn.it
[2] Dipartimento di Fisica, Università di Firenze, Sesto Fiorentino (FI), Italy
[3] Istituto Nazionale di Fisica Nucleare, Sezione di Torino, Turin, Italy
[4] Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Perugia, Perugia, Italy

Abstract. In this paper we present dGanX, a Python library to generate a large number of X-ray Fluorescence Macro maps (MA-XRF) from coloured RGB images, in a distributed manner across multiple computing nodes. To generate a synthetic MA-XRF, a Monte Carlo method is used, where each MA-XRF pixel signal is sampled out of an XRF signal probability function. Such probability function is computed using a database of couples (pigment characteristic XRF signal, RGB), by a weighted sum of such pigment XRF spectra by proximity of the image RGB to the pigment characteristic RGB. To distribute the synthetic MA-XRF generation of multiple images, the code was refactored such that it can be distributed on a Kubernetes cluster via Dask. The code is open source and freely available.

Keywords: Simulation methods and programs · Analysis and statistical methods · Computing · X-ray fluorescence (XRF) systems

1 Introduction

In the application of statistical and machine learning methods to data obtained from spectral imaging of cultural heritage (CH), one of the main issues is the lack of huge amounts of available data (see, e.g., [6,17], and references therein). Nevertheless, in the last decade, we have witnessed a truly remarkable rise of the applications of such methods in the CH sector, also in the realm of nuclear physics applied to CH diagnostics and imaging These techniques are being increasingly utilized to support researchers in interpreting data obtained through non-invasive testing approaches [20], such as hyperspectral imaging [18,32,35], neutron radiography, gamma spectroscopy, and X-ray-based imaging modalities [19,30]. Such algorithms can reveal correlations and hidden structures in the

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16758, pp. 70–83, 2027.
https://doi.org/10.1007/978-3-032-30497-1_5

datasets, providing valuable clues about an artifact's material makeup [24], dating, and preservation status [11,17,34][1].

One of the most notable nuclear methodologies applied to the field of Cultural Heritage is X-Ray Fluorescence (XRF) spectroscopy, particularly in its macro-scale mapping variant (MA-XRF). In Macro X-Ray Fluorescence (MA-XRF) imaging, the system generates a data cube in which each pixel is associated with a spectrum containing fluorescence peaks indicative of the elemental makeup of pigments within the pictorial layers. Due to the vast quantity of pixel-wise XRF spectra compared to the limited variety of pigment palettes, MA-XRF data cubes provide a particularly suitable structure for implementing unsupervised statistical learning techniques [30].

An intriguing application of Computer Vision on such datasets is the so-called *virtual colouring*, *i.e.* the creation of a synthetic RGB image from the MA-XRF data cube of an artwork under examination. This can have practical implications in the domain of conservation, as MA-XRF can detect elemental signals, even from damaged or aged surfaces (paintings or frescoes alike) [1,13,27], and allows analysis of otherwise obscured features, including underdrawings, pentimenti [10,29], areas of previous restoration, or pigment loss [12,22]. Such computer-aided workflows offer conservators and heritage specialists a new channel of interpretation by providing a complementary visual output (*i.e.*, the rendered RGB reconstruction).

Although few attempts have been conducted [3–6], yet, the availability of high-quality datasets for training these models is often constrained by several factors: sluggish acquisition speeds due to hardware limitations, legal and Intellectual Property (IP) restrictions, limited access to test artifacts, the scarcity and expense of suitable instrumentation, and the necessity for highly skilled operators. Moreover, unlike in more standardized scientific domains, data harmonization remains a challenge, given the varied custom setups of acquisition systems, heterogeneous measurement conditions, and detector types.

This justifies the emphasis put on the creation of ad hoc *synthetic* MA-XRF dataset [17]. In this paper, we present `dGanX`, a Python library for creating synthetic dataset of MA-XRF images in a distributed manner, starting from

1. A (set of) RGB image(s); and
2. A Database of pigmentsl.

Such database must have

a. A characteristic XRF signal;
b. A characteristic, 8-bit, RGB colour.

`dGanX` is based on, and it is an evolution of, `ganX`[2], and it will use (b) to associate pigment(s) to RGB pixels, and (a) as a probability distribution to

[1] For additional machine learning implementations within Cultural Heritage, refer to [14,21], and references therein.
[2] https://pypi.org/project/ganx/, https://github.com/androbomb/ganX.

generate the MA-XRF data cube via Monte Carlo methods. The source code is freely available[3].

The contribution of this paper is:

- To present the code for the distributed generation of synthetic MA-XRF data-cubes;
- Present an example of integration of distributed ML inference on synthetic MA-XRF data *on the fly*, meaning at generation time;
- Present an example of a comparison of a real and a generated synthetic data-cube.

The paper is organised as follows: in Sect. 2, we discuss the algorithm to generate MA-XRF datacubes out of RGB images and a pigment palette; in Sect. 3 we discuss how the distributed generation of multiple MA-XRF on a `Dask` cluster is achieved; in Sect. 4, we provide an example of usage of `dGanX`; finally, in Sect. 5, we report the conclusions and future outlook.

2 How to Generate MA-XRF from RGB

Inspired by the work done in [17], where the authors generate a synthetic dataset of single-point XRF histogram by using the Fundamental Parameters (FP) method, a physical model based on Sherman's equations for generating XRF spectra of a sample of known composition [28], we started the development of a simple Python package capable of creating MA-XRF data cubes starting from a RGB image and a database of pigments' XRF signal. The data generation algorithm comprises two main parts:

1. A RGB clustering of the input image(s), performed with an iterative K-Means, to reduce the colour noise, thus *quantising* the RGB image;
2. A Monte Carlo method to generate a XRF histogram of counts for each pixel, after having unsupervisedly associated a (set of) pigment(s) to each RGB pixel.

To perform step 2. above, we compute (pixel-by-pixel) a set of similarity measures in RGB spaces, between the (quantised) RGB and the database indexed pigments' characteristic RGB; then, a hard thresholding is performed, to (unsupervisedly) reduce the number of pigments composing a color.

The full pseudocode of the MA-XRF data generation is reported in Algorithm 1.

The actual code implementation in Python leverages the `NumPy` [16,31] *vectorization* to speed up the computation.

Note on [17]*:* Notice that it is possible to use the FP approach of [17] to build the characteristic pigments' XRF signal; this implies that it is straightforward to use the `dGanX` library to leverage the synthetic data generation of [17] from single-point XRF to MA-XRF data cubes.

[3] https://baltig.infn.it/bombini/fastextendedvision-dask.

Algorithm 1. Generate MA-XRF

Require: RGBclustering, distrToXrf, colorSimilarity ▷ Import relevant methods
procedure generateXRF(pigments_dict_data = $\mathcal{D}_{color,hist}$, rgb_img = R)
  $R \leftarrow$ RGBclustering(R) ▷ Quantise RGB
  init X ▷ Initialise empty XRF data cube
  for $r \in R$ **do** ▷ Iterate over clusters
    init d ▷ Initialise empty distribution
    for $(c, h) \in \mathcal{D}$ **do** ▷ Iterate over palette
      $\alpha \leftarrow$ colorSimilarity(c, r) ▷ Compute similarity scores
      if $\alpha \geq \alpha_{th}$ **then**
        $d \leftarrow d + \alpha \cdot h$ ▷ Sum thresholded distribution
    $d \leftarrow d/||d||$ ▷ Normalise distribution
    $X[\text{idx}(X) == r] \leftarrow$ distrToXrf(d) ▷ Use Monte Carlo method only on the Cluster's pixel
  return X

On the other hand, `dGanX` can be used out of *real* data, comprising measurement whose source origin is complicated enough that no analytic method (such as the FP method) can be used to generate the XRF signal. This latter situation is quite common in the field of Cultural Heritage applications.

Additional note: Notice also that the spectral signal is not limited to be an XRF characteristic signal; it is possible to generate data cubes out of other spectroscopy techniques, not even in the X-Ray range, such as *Fourier Transform InfraRed spectroscopy* (FTIR), HyperSpectral Imaging (HSI), Particle Induced X-Ray Emission (PIXE), etc., just to name a few.

3 Distributed Generation of MA-XRF with `Dask`

Since the algorithm converts RGB images into MA-XRF, one may want to start generating a *huge* dataset of MA-XRF starting from, for example, *all* the pictures in PNG/JPG format representing pictorial artworks contained in WikiData, in order to have a huge dataset for computer vision application, as done, for example, in [6].

By scraping the WikiData SPARQL endpoint[4], one may get over 230.000 CC0 RGB images, which can be transformed, one by one, in MA-XRF using algorithm 1. The algorithm by `dGanX` simply:

1. Iterates over all RGB images;
2. Launches a single RGB-to-XRF conversion for each of them (via Algorithm 1);
3. After that, stores the generated MA-XRF image on Disk.

To speed up the generation of MA-XRF images, we can simply process each RGB-to-MA-XRF *in parallel*, distributing it not only on each independent CPU thread, but also on different CPUs on multiple computing nodes. Since each generation process is completely independent from the others, it can be massively

[4] https://query.wikidata.org/.

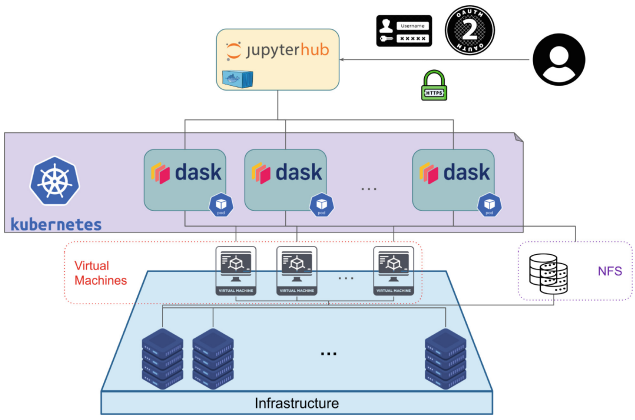


Fig. 1. Visual representation of the cloud architecture for the distributed computing setting.

parallelised, effectively reducing the generation time of the whole dataset by a factor of N, where $N = \sum_{n=1}^{N_{\text{nodes}}} t_i$ is the total number of independent threads t_i on all the computing nodes.

3.1 Distributing MA-XRF Generation Using Dask on a Kubernetes Cluster

To design, develop, test, and deploy the distributed computing approach of `dGanX` we used a Kubernetes[5] (k8s) cluster, built on top of multiple Virtual Machines (VMs). Both `Dask` [26], as well as JupyterHub[6] are executed as kubernetes applications through pods. Each pod mounts both ephemeral File System (FS) and a persistent FS of small size ($\sim O(10\text{Gb})$), designed to host the code during development. Moreover, a shared Network File System (NFS), of bigger size ($\sim O(1\text{Tb})$), has been configured in order to manage long term storage and data sharing between nodes.

From the user perspective, after an Authentication/Authorisation process handled by INDIGO-IAM[7], based on the OAuth2 protocol, the user accessess the JupyterHUB spawning page, where it is possible to pick the docker image[8] to be deployed, containing the necessary libraries. Once the JupyterHUB successfully spawns the notebook (JupyterLAB), the user can deploy and manage several Dask Clusters on Kubernetes nodes via the Dask Operator[9]. This can be done either via the embedded Dask extension in the JupyterHUB page, or via Python

[5] https://kubernetes.io/.
[6] https://jupyter.org/hub.
[7] https://github.com/indigo-iam/iam.
[8] https://www.docker.com/.
[9] https://kubernetes.dask.org/en/latest/operator_kubecluster.html.

using the KubeCluster interface, offered in the dask-kubernetes package[10]. A visual abstract of the infrastructure is reported in Fig. 1.

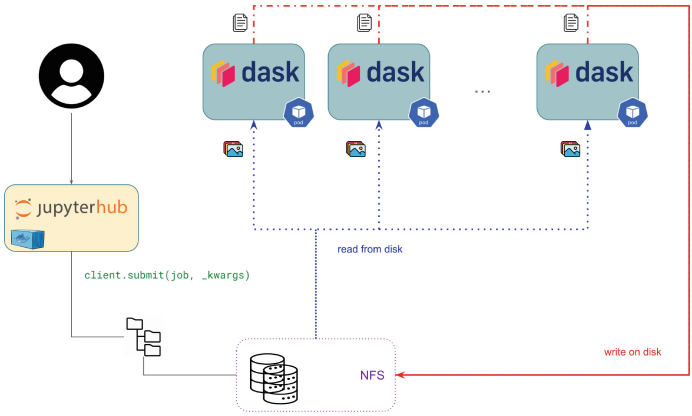


Fig. 2. Visual representation of the multi-node parallelisation with Dask. The user, via the web UI or the CLI in the JupyterHub web page, can launch the job, iterating over all RGB images in the NFS; the master node (the one running the JupyterHub) dispatch the jobs to each available node, in parallel, signaling which file has to read on the NFS. The node opens the RGB file, perform Algorithm 1, and then store the generated MA-XRF back on a specified location, with the same name as the RGB file but in HDF5 extension.

Nevertheless, dGanX only assumes the existence of a Dask cluster (not necessarily a KubeCluster); the needed set up is that the code has to be available on the shared NFS, as are the RGB files, so that each node may have access to the required Python codes. After that, via a simple Command Line Interface (CLI) command, it is possible to launch the distributed generation pipeline which, at its core, is simply a parallel distribution of jobs, as reported in Algorithm 2:

Algorithm 2. Distribute Generation

Require: generateXRF, KubeCluster ▷ Import relevant methods
procedure distributeGeneration(pigments_dict_data = $\mathcal{D}_{color,hist}$, path_to_imgs = P)
  init Client ▷ Initialise KubeClient
  for $p \in P$ **do** ▷ Iterate over RGB files
    if !notGeneratedYet(p) **then** ▷ Check if file has been already processed
      client.submit(generateXRF , { p, $\mathcal{D}_{color,hist}$ }) ▷ Submit Job

From the master node, the user can launch via the CLI the distributed generating pipeline,

```
$ python3 create_2D_dataset_dask_distributed.py [ARGS]
```

[10] https://pypi.org/project/dask-kubernetes/, https://github.com/dask/dask-kubernetes.

and pass a set of values as arguments to specialise the generation, such as the FS path to the RGB folder, `-rootdir`, the path to store the produced MA-XRF datacubes as `HDF5`, `-storedir`, as well as the parameters describing the Algorithm 1, as the average number of counts per energy channel, `-extractionsize`, or the minimum or maximum number of RGB cluster to extract, `-clustlow` and `-clusthigh`.

After the submission of the CLI command, the master node starts iterating over the files in the NFS, packaging the necessary arguments to be passed to each node, and then submit a job to each available node; when the number of nodes are exhausted, the master node awaits the first node to complete the task, and then dispatches to it the next job, until completion. For a visual representation, see Fig. 2.

A peculiarity is that the code has to be on the NFS, since, to reduce the network communication payload, the relevant part of the codes are put in importable packages, which have to be available to all nodes. Each computing node is thus instructed to read, load, and execute the relevant code (with the passed *keywords*) *locally*, from a protected *read-only* section of the FS (the one containing the RGB images and the code), and store the result in a *writable* part of the NFS. The only relevant part of the infrastructure is that the node has (at least) reading privilege to the former, and writing privilege to the latter. The two can be different NFS, or any other shared FS, such as mounted `WebDav` FS (via, e.g., `davfs2`[11]).

3.2 Adding a Machine Learning Distributed Inference with `UMAP`

Sometimes, while performing analysis on MA-XRF datacubes, it may be helpful to train a *feature extractor* machine learning model to reduce the dimensionality of the data [6,7]. To show how `dGanX` can easily integrate a similar feature, we trained a `UMAP` model [23], and added it in the generation pipeline.

`UMAP` is a *manifold learning* algorithm, capable of extracting *non-linear* relation among features, and extract low-dimensional representation of data out of them. It can be used, similarly to what has been done in [6,7], as a *feature extractor* to perform *unsupervised domain adaptation*; i.e., we can train a UMAP model on the spectra of a *real* datacube, and use it to extract features from the *synthetic* MA-XRF spectra, so that it extracts features in a *domain adapted* manner. In this way, the trained `UMAP` model extracts low-dimensional representation of MA-XRF spectra which are relevant in the *domain* of the real MA-XRF datacube.

In Fig. 3, the inference of the UMAP model is shown. The `UMAP` model has been trained on the spectra contained in a single, real, MA-XRF datacube, labelled `ft` (*fine tuning*); the model learns how to map the $D = 512$ dimensional spectra to a 3-dimensional space, the *latent* space. The model was trained using the Python package of [23], `umap-learn`[12], using the values `n_neighbors`=15, `min_dist`=0.1, `metric`=`'euclidean'`, `spread`=1.0, `local_connectivity`=1.0.

[11] https://savannah.nongnu.org/projects/davfs2.
[12] https://umap-learn.readthedocs.io/en/latest/basic_usage.html.

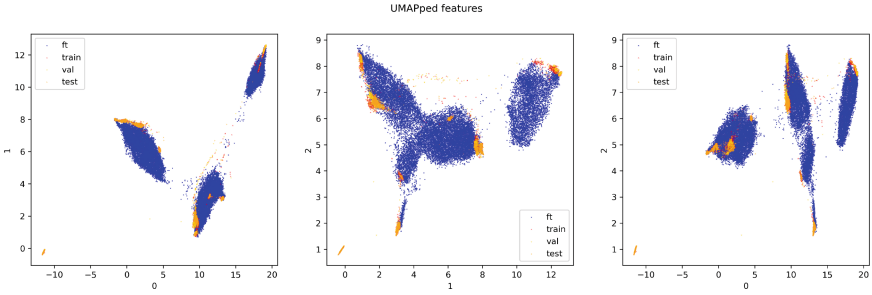


Fig. 3. Example of the action of the `UMAP` model. The three images represents the 2D projections of the 3D reduced space scatter plot. The blue dots represents the mapped low-dimensional points used for training, coming from the real datacube (where `ft` stands for *fine tuning*); the red, yellow and orange dots, instead, represents the inference of the trained `UMAP` model on different spectra, coming from the train, validation and test set (where, here, train, test and validation set are simply three different group of MA-XRF images). (Color figure online)

Now, we used the trained model to perform the inference on a set of random spectra coming from a different set of generated MA-XRF, that we call train, validation, test, since they can be used as train, validation and test in a computer vision task. We see that the low-dimensional features of these spectra, which have never been seen during training of the UMAP model, are mapped in a distribution which resembles the distribution of the real data, showing that the low-dimensional latent space provides a form of *unsupervised domain adaptation*.

In the distributed computing approach, each node, after having generated the MA-XRF datacube out of the RGB image, and stored it on NFS, applies the trained UMAP model to reduce its dimensionality, and store the dimensionally reduced datacube in the same `HDF5` file, as a different `HDF5Dataset`.

This is simply accomplished by a slight modification of Algorithm 2:

Notice that the UMAP model is *pre-trained*, i.e. it is not trained during the job, but only used for *inference*. Additionally, the model has to be loaded by each node *independently*, and thus has to be accessible from each nodes, i.e. has to be stored in the shared NFS.

Algorithm 3. Distribute Generation and `UMAP` inference

Require: generateXRF, KubeCluster, model ▷ Import relevant methods
1: **procedure** GenerateAndInference(pigments_dict_data = $\mathcal{D}_{color,hist}$, path_to_imgs = P)
2: `load`(model) ▷ Load model from Disk
3: X = generateXRF ({ p, $\mathcal{D}_{color,hist}$ }) ▷ Generate MA-XRF
4: Y = model (X) ▷ Perform Model inference
5: `StoreHDF5` (X, Y) ▷ Store results
6: **procedure** distributeGenerationAndInference(pigments_dict_data = $\mathcal{D}_{color,hist}$, path_to_imgs = P)
7: init Client ▷ Initialise KubeClient
8: **for** $p \in P$ **do** ▷ Iterate over RGB files
9: **if** !NotGeneratedYet(p) **then** ▷ Check if file has been already processed
10: client.submit(GenerateAndInference , { p, $\mathcal{D}_{color,hist}$ }) ▷ Submit Job

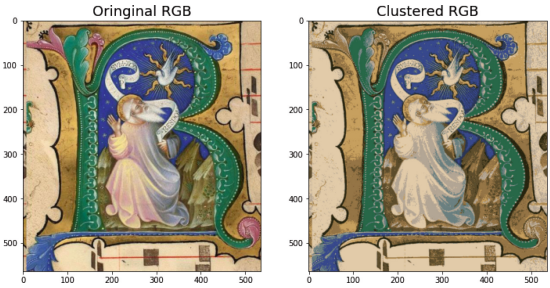


Fig. 4. Left: RGB image of Antiphonary M, folio 32. Right: clustered RGB with iterative KMeans ($K = 11$).

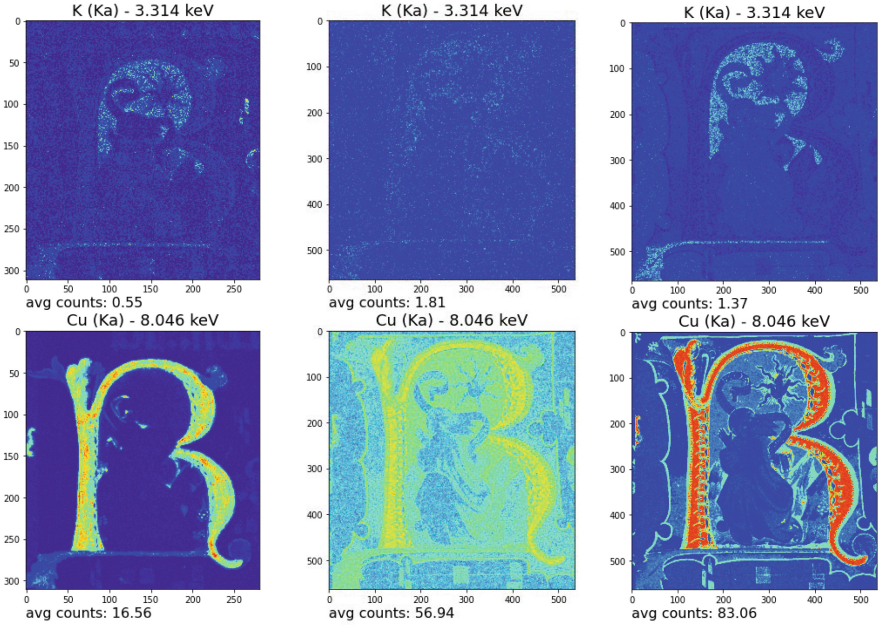
Using Other Models: we have presented the result of distributed ML approach using a (trained) UMAP model; nevertheless, it is easy to generalise this approach to include more complex ML models, such as a Deep Variational Embedding auto-encoder, such as the one trained on MA-XRF data in [7].

4 Example: Generation from a Real Artwork

To illustrate the usage of `dGanX`, we built a small pigment database comprising nine pigments (lead red, vermillion, organic red dye, lead tin yellow, ultramarine, azurite, indigo, malachite, chalk white). The XRF signals were taken from the open database INFRA-ART [9], while the RGB colours were obtained from standard references.

We then generated a synthetic MA-XRF dataset starting from the RGB image of the Antiphonary M, folio 32, an illuminated manuscript studied in [25]. The RGB image is shown in Fig. 4 (left). After an iterative KMeans clustering (right), the number of distinct colour regions is reduced, improving the stability of the subsequent pigment assignment.

We generated two synthetic cubes with the same parameters except for the similarity threshold α_{th}, set to 0.2 and 0.6. Figure 5 compares the elemental maps of potassium (K (Kα)) and calcium (Ca (Kα)) obtained from the real MA-XRF data and from the two synthetic versions. A lower threshold allows more pigments to contribute to each pixel, resulting in a smoother distribution; a higher threshold yields sharper, more localized maps.



(a) Real MA-XRF (b) Synthetic $\alpha_{\text{th}} = 0.2$ (c) Synthetic $\alpha_{\text{th}} = 0.6$

Fig. 5. Comparison of elemental maps for K (Kα) and Cu (Kα).

A quantitative comparison can be performed on the integrated histogram of all XRF counts (Fig. 6). The Bhattacharyya coefficient (similarity) and the Structural Similarity Index Measure (SSIM) [33] computed on the flattened hypercubes are reported in Table 1. Both metrics indicate that the synthetic data capture the main features of the real distribution, with the lower threshold producing slightly closer agreement.

Table 1. Similarity metrics between real and synthetic MA-XRF data.

Metric	$\alpha_{\text{th}} = 0.2$	$\alpha_{\text{th}} = 0.6$
Bhattacharyya coefficient	0.8529	0.8589
SSIM	0.3491	0.3317

This example demonstrates the ability of `dGanX` to produce realistic MA-XRF data cubes from RGB images, while highlighting the impact of the similarity threshold on the output.

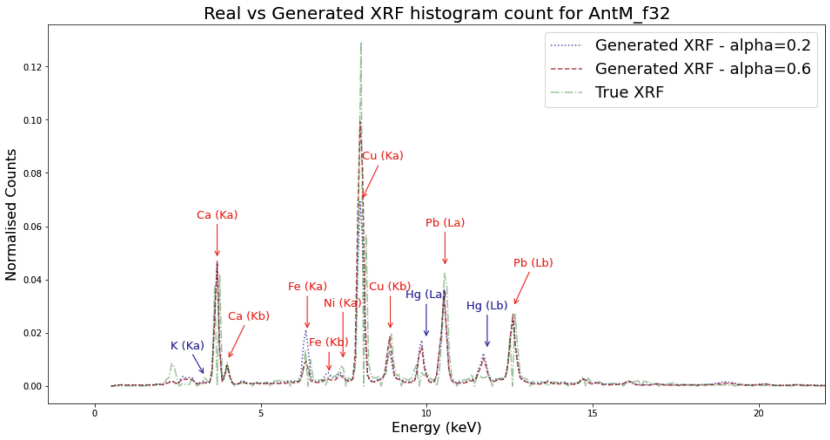


Fig. 6. Comparison of total XRF count histograms (normalised).

5 Conclusions

In this paper we have presented `dGanX`, a Python package offering methods to generate artificial new MA-XRF data cubes out of an RGB image, in a distributed manner. This package may be helpful in the synthetic dataset generation for applications of statistical and deep learning methods in the field of MA-XRF analysis, especially in (but not limited to) the field of Cultural Heritage.

The possibility of distributing the dataset generation on multiple threads on multiple computing nodes may be helpful in reducing the time needed for synthetic data generation of huge datasets, like the one needed in computer vision tasks requiring multiple synthetic datasets, like [6].

In the future, we plan to explore other approaches to distributed computing, focusing on the possibility to offload the computation on HPC clusters, via a different approach through containerisation (e.g., via *Apptainer*[13] or *Docker* containers) of the computing environment. Additionally, we will work on improving the MA-XRF generation pipeline to extend the flexibility of the package, by adding additional features, such as the possibility to generate MA-XRF data cubes without a fixed amount of signal per-pixel, but allowing for, e.g. luminosity-based number of counts. Also, we are exploring the possibility of adding Poissonian noise, as well as other noise sources.

Acknowledgments. To conduct the work presented in this contribution, we resorted to the cloud computing Software-as-a-Service infrastructure made available by the ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, and by AI_INFN initiative [2], as well as some servers made available by INFN-CHNet [15], the network devoted to the application of nuclear techniques to Cultural Heritage.

[13] https://apptainer.org/.

We would also like to thank the authors of the site https://infraart.inoe.ro/ [8], and WikiData (https://www.wikidata.org) and WikiMedia commons (https://commons.wikimedia.org).

We thank the authors of [25] for sharing the raw data used in Sect. 4. The analysis conducted in the aforementioned paper on the Anphitionary M belonging to the collection of San Giorgio Maggiore was funded by the Abbey of San Giorgio Maggiore.

The work of AB and TT was funded by Progetto ICSC - Spoke 2 - Codice CN00000013 - CUP I53C21000340006 - Missione 4 Istruzione e ricerca - Componente 2 Dalla ricerca all'impresa – Investimento 1.4.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alberti, R., et al.: Handheld XRF and Raman equipment for the in situ investigation of Roman finds in the Villa dei Quintili (Rome, Italy). J. Anal. At. Spectrom. **32**, 117–129 (2017). https://doi.org/10.1039/C6JA00249H, http://dx.doi.org/10.1039/C6JA00249H
2. Anderlini, L., et al.: Supporting the development of machine learning for fundamental science in a federated cloud with the AI_infn platform. arXiv preprint arXiv:2502.21266 (2025)
3. Bombini, A., Bofías, F.G.A., Ruberto, C., Taccetti, F.: A cloud-native application for digital restoration of Cultural Heritage using nuclear imaging: THESPIAN-XRF. Rendiconti Lincei. Scienze Fisiche e Naturali **34**, 867–887 (2023). https://doi.org/10.1007/s12210-023-01174-0, https://api.semanticscholar.org/CorpusID:259707375
4. Bombini, A., Anderlini, L., dell'Agnello, L., Giacomini, F., Ruberto, C., Taccetti, F.: Hyperparameter optimisation of artificial intelligence for digital REStoration of cultural heritages (AIRES-CH) models. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds.) ICCSA 2022. LNCS, vol. 13377, pp. 91–106. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-10536-4_7
5. Bombini, A., Anderlini, L., dell'Agnello, L., Giacomini, F., Ruberto, C., Taccetti, F.: The AIRES-CH project: artificial intelligence for digital REStoration of cultural heritages using nuclear imaging and multidimensional adversarial neural networks. In: Sclaroff, S., Distante, C., Leo, M., Farinella, G.M., Tombari, F. (eds.) ICIAP 2022. LNCS, vol. 13231, pp. 685–700. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06427-2_57
6. Bombini, A., Bofías, F.G.A., Giambi, F., Ruberto, C.: Towards virtual painting recolouring using vision transformer on x-ray fluorescence datacubes. Mach. Learn. Sci. Technol. **6**(1), 015058 (2025). https://doi.org/10.1088/2632-2153/adb937
7. Bombini, A., Bofías, F.G.A., Bracci, C., Ginolfi, M., Ruberto, C.: Datacube segmentation via deep spectral clustering. Mach. Learn.: Sci. Technol. (2024). https://doi.org/10.1088/2632-2153/ad622f, http://iopscience.iop.org/article/10.1088/2632-2153/ad622f
8. Cortea, I.M., Chiroşca, A., Angheluţă, L.M., Seriţan, G.: Infra-art: an open access spectral library of art-related materials as a digital support tool for cultural heritage science. J. Comput. Cult. Herit. **16**(2) (2023). https://doi.org/10.1145/3593427

9. Cortea, I.M., Chiroşca, L.M., Angheluţă, L.M., Seriţan, G.: INFRA-ART: an open access spectral library of art-related materials as a digital support tool for cultural heritage science. ACM J. Comput. Cult. Herit. **16**(2) (2023)
10. Dik, J., Janssens, K., Van Der Snickt, G., van der Loeff, L., Rickers, K., Cotte, M.: Visualization of a lost painting by Vincent van Gogh using synchrotron radiation based x-ray fluorescence elemental mapping. Anal. Chem. **80**(16), 6436–6442 (2008). https://doi.org/10.1021/ac800965g
11. Dirks, M., Poole, D.: Auto-encoder neural network incorporating x-ray fluorescence fundamental parameters with machine learning. X-Ray Spectrometry **52**(3), 142–150 (2023). https://doi.org/10.1002/xrs.3340, https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/xrs.3340
12. Dredge, P., Ives, S., Howard, D.L., Spiers, K.M., Yip, A., Kenderdine, S.: Mapping henry: synchrotron-sourced x-ray fluorescence mapping and ultra-high-definition scanning of an early tudor portrait of henry viii. Appl. Phys. A **121**, 789–800 (2015)
13. Fiorillo, F., Fiorentino, S., Montanari, M., Roversi Monaco, C., Del Bianco, A., Vandini, M.: Learning from the past, intervening in the present: the role of conservation science in the challenging restoration of the wall painting marriage at Cana by Luca Longhi (Ravenna, Italy). Herit. Sci. **8**(1), 10 (2020). https://doi.org/10.1186/s40494-020-0354-y
14. Fiorucci, M., Khoroshiltseva, M., Pontil, M., Traviglia, A., Del Bue, A., James, S.: Machine learning for cultural heritage: a survey. Pattern Recogn. Lett. **133**, 102–108 (2020). https://doi.org/10.1016/j.patrec.2020.02.017
15. Giuntini, L., et al.: Detectors and cultural heritage: the INFN-CHNet experience. Appl. Sci. **11**(8) (2021). https://doi.org/10.3390/app11083462
16. Harris, C.R., et al.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2
17. Jones, C., Daly, N.S., Higgitt, C., Rodrigues, M.R.D.: Neural network-based classification of x-ray fluorescence spectra of artists' pigments: an approach leveraging a synthetic dataset created using the fundamental parameters method. Herit. Science **10**(1), 88 (2022). https://doi.org/10.1186/s40494-022-00716-3
18. Kleynhans, T., Schmidt Patterson, C.M., Dooley, K.A., Messinger, D.W., Delaney, J.K.: An alternative approach to mapping pigments in paintings with hyperspectral reflectance image cubes using artificial intelligence. Herit. Sci. **8**(1), 84 (2020). https://doi.org/10.1186/s40494-020-00427-7
19. Kogou, S., Lee, L., Shahtahmassebi, G., Liang, H.: A new approach to the interpretation of XRF spectral imaging data using neural networks. X-Ray Spectrometry **50**(4) (2020). https://doi.org/10.1002/xrs.3188
20. Licciardi, G.A., Del Frate, F.: Pixel unmixing in hyperspectral data by means of neural networks. IEEE Trans. Geosci. Remote Sens. **49**(11), 4163–4172 (2011). https://doi.org/10.1109/TGRS.2011.2160950
21. Liu, L., et al.: Neural networks for hyperspectral imaging of historical paintings: a practical review. Sensors **23**(5) (2023). https://doi.org/10.3390/s23052419, https://www.mdpi.com/1424-8220/23/5/2419
22. Martins, A., et al.: Jackson Pollock's number 1a, 1948: a non-invasive study using macro-x-ray fluorescence mapping (ma-xrf) and multivariate curve resolution-alternating least squares (mcr-als) analysis. Herit. Sci. **4**, 1–13 (2016)
23. McInnes, L., Healy, J., Melville, J.: Umap: uniform manifold approximation and projection for dimension reduction (2020)

24. Preisler, Z., Andolina, R., Busacca, A., Caliri, C., Miliani, C., Romano, F.P.: Deep learning for enhanced spectral analysis of ma-xrf datasets of paintings. Sci. Adv. **10** (2024). https://api.semanticscholar.org/CorpusID:272881694
25. Ricciardi, P., Mazzinghi, A., Legnaioli, S., Ruberto, C., Castelli, L.: The Choir books of San Giorgio Maggiore in Venice: results of in depth non-invasive analyses. Heritage **2**(2), 1684–1701 (2019). https://doi.org/10.3390/heritage2020103
26. Rocklin, M.: Dask: parallel computation with blocked algorithms and task scheduling. In: Proceedings of the 14th Python in Science Conference, pp. 130–136, Citeseer (2015)
27. Ruberto, C.: The mission of the INFN-cultural heritage network: the multifaceted example of the macro-xrf scanner experience. Rendiconti Lincei. Scienze Fisiche e Naturali **34**(3), 889–906 (2023). https://doi.org/10.1007/s12210-023-01175-z
28. Sherman, J.: The theoretical derivation of fluorescent x-ray intensities from mixtures. Spectrochim. Acta **7**, 283–306 (1955)
29. Van der Snickt, G., et al.: In situ macro x-ray fluorescence (ma-xrf) scanning as a non-invasive tool to probe for subsurface modifications in paintings by p.p. rubens. Microchem. J. **138**, 238–245 (2018). https://doi.org/10.1016/j.microc.2018.01.019, https://www.sciencedirect.com/science/article/pii/S0026265X17308779
30. Vermeulen, M., et al.: XRFast a new software package for processing of MA-XRF datasets using machine learning. J. Anal. At. Spectrom. **37**, 2130–2143 (2022). https://doi.org/10.1039/D2JA00114D
31. van der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation. Comput. Sci. Eng. **13**(2), 22–30 (2011). https://doi.org/10.1109/mcse.2011.37
32. Wang, M., Zhao, M., Chen, J., Rahardja, S.: Nonlinear unmixing of hyperspectral data via deep autoencoder networks. IEEE Geosci. Remote Sens. Lett. **16**(9), 1467–1471 (2019). https://doi.org/10.1109/LGRS.2019.2900733
33. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861
34. Xu, B.J., et al.: Can deep learning assist automatic identification of layered pigments from xrf data? (2022). https://doi.org/10.1039/D2JA00246A
35. Zhang, X., Sun, Y., Zhang, J., Wu, P., Jiao, L.: Hyperspectral unmixing via deep convolutional neural networks. IEEE Geosci. Remote Sens. Lett. **15**(11), 1755–1759 (2018). https://doi.org/10.1109/LGRS.2018.2857804