Journal of Vacuum Science and Technology A

RESEARCH ARTICLE | JUNE 01 2026

Considerations for implementing real-time machine learning tools to evaluate ToF-SIMS data

Brian Oslinker ; Wil Gardner ; Sarah E. Bamford ; See Yoong Wong ; John A. Webb ; Davide Ballabio ; Thomas M. Kohl ; Paul J. Pigram

J. Vac. Sci. Technol. A 44, 043204 (2026)
https://doi.org/10.1116/6.0005335

Connected Content

A companion article has been published: Optimizing ToF-SIMS data analysis with survey-then-target workflow

Articles You May Be Interested In

Effect of data preprocessing and machine learning hyperparameters on mass spectrometry imaging models

J. Vac. Sci. Technol. A (September 2023)

JVST A
Journal of Vacuum Science & Technology A

ARTICLE

pubs.aip.org/avs/jva

Considerations for implementing real-time machine learning tools to evaluate ToF-SIMS data

Cite as: J. Vac. Sci. Technol. A 44, 043204 (2026); doi: 10.1116/6.0005335
Submitted: 19 January 2026 · Accepted: 23 April 2026 ·
Published Online: 1 June 2026

Brian Oslinker,[1] Wil Gardner,[1] Sarah E. Bamford,[1] See Yoong Wong,[1] John A. Webb,[2] Davide Ballabio,[3] Thomas M. Kohl,[4] and Paul J. Pigram[1,a)]

AFFILIATIONS

[1]Centre for Materials and Surface Science and Department of Mathematical and Physical Sciences, La Trobe University, Bundoora 3086, Victoria, Australia
[2]School of Agriculture, Biomedicine and Environment, La Trobe University, Bundoora 3086, Victoria, Australia
[3]Milano Chemometrics and QSAR Research Group, Department of Earth and Environmental Sciences, University of Milano-Bicocca, Piazza Della Scienza 1, Milano 20126, Italy
[4]CSIRO Manufacturing, Clayton 3168, Victoria, Australia

[a)]**Author to whom correspondence should be addressed:** p.pigram@latrobe.edu.au

ABSTRACT

Time-of-flight secondary ion mass spectrometry (ToF-SIMS) produces complex, information-rich, high-dimensional chemical data sets that can be challenging to analyze and interpret. Computational methods for dimensionality reduction offer effective pathways for addressing these issues. This work offers guidance, exemplars, and benchmarking options for optimizing machine learning computation that are specifically relevant to the investigation of large scale ToF-SIMS data sets and generally applicable to other machine learning applications. The guidance is formed around the self-organizing map with relational perspective mapping (SOM-RPM) MATLAB toolbox developed by our group, as a practical example of optimization, computation speed up, and related efficiency improvements. Optimization approaches considered include processor selection and methods of deployment, and cleanup of code to reduce duplicate calculations. This work explores the practical trade-offs of using double precision floating point arithmetic on execution speeds, in particular, for parallel calculations on the GPU, in comparison with single precision. The interaction between the inherent spectral and signal-to-noise characteristics of the ToF-SIMS data and the floating-point precision—in terms of machine learning model quality and convergence—is considered. An illustrative case study of a mineral thin section is presented using ToF-SIMS and SOM-RPM for the rapid identification of regions of interest to guide subsequent high-resolution scans. We have documented speed improvements (of code execution time) up to two orders of magnitude in our SOM-RPM toolbox. These improvements not only reduce computational latency but also open a feasible trajectory for real-time machine learning deployments in surface analysis workflows.

I. INTRODUCTION

Time-of-flight secondary ion mass spectrometry (ToF-SIMS) is a powerful surface analysis technique with the capacity to generate information-rich spectra, hyperspectral maps, and depth profiles. ToF-SIMS combines a small spot size with the acquisition of a complete mass spectrum for each pixel or voxel analyzed.[1,2]

The scale and complexity of ToF-SIMS data sets pose significant challenges to interpretation. Matrix effects, variable sputter yields, differences in ion potentials, and fragmentation, among other factors, can lead to nonlinearity in the data,[2,3] further complicating analysis.

Computational techniques have become vital for analyzing complex imaging mass spectrometry data. Unsupervised machine

learning (ML) has emerged as a tool with great potential. Numerous implementations have been described and reviewed in detail in the literature.[4,5] Commonly encountered computational methods include principal component analysis (PCA)[6,7] (often used as a preprocessing step prior to other methods), K-means clustering,[8–10] t-distributed stochastic neighbor embedding (t-SNE),[11,12] uniform manifold approximation and projection (UMAP),[13–15] and self-organizing maps (SOMs).[5,12,16–18] This is by no means an exhaustive list. More recent work has continued to demonstrate the utility of SOMs for mass spectrometry data analysis.[19]

Due to the expansive landscape of available computational techniques and the highly specific requirements of individual researchers, out-of-the-box solutions at the software level frequently do not exist or are not easily accessible. This places the burden on domain experts themselves to adapt and enhance existing software to meet specific analytical needs, or to create bespoke code or computational platform environments entirely from scratch. In these instances, the priority is typically and understandably to ensure the underlying mathematics and statistics is correct. Significant expertise in computer science and software engineering is required to produce high quality code consistent with professional software design standards. Experimental and evolving code versions may exhibit inefficiencies and suboptimal runtime.

Suboptimal code runtime may be unimportant for many data sets, particularly in cases with smaller data volumes. However, barriers rapidly emerge when considering large scale data sets and real-time (or close to real-time) data analysis, though some efforts have been made to address this.[20] The objective of real-time data analysis is to provide actionable insights live at the instrument interface as the data are being acquired. Such capabilities will form a core element in future artificial intelligence (AI)-based instrument control platforms[21–23] and are essential when, for example, data volumes and rates of production are so high that no practical storage solution exists[20,24] (i.e., on the fly analyses). There is a need to describe and implement readily accessible steps, modifications and design considerations in methodology that can significantly improve computational efficiency, allowing more efficient use of available computing resources with minimal effort.

In this work, we focus on the freely available SOM-RPM Toolbox for MATLAB,[25] which provides a solution for investigating complex, large scale, multidimensional mass spectral data sets. We investigate systematic improvements and code implementations to deliver a substantial improvement in the efficiency and runtime of the toolbox. Improvements in computational speed up to two orders of magnitude are demonstrated, measured against benchmarks. We discuss broader concepts around implementing code to best utilize available computing resources, including floating-point precision, central processing unit (CPU) and graphics processing unit (GPU) utilization, as well as code considerations to avoid redundant computation. Finally, we demonstrate the enhanced capability of the toolbox to support real-time decision-making and on-the-fly spectral interpretation with a ToF-SIMS case study of a mineral sample. This work aims to provide a generally applicable resource for researchers, to assist in the development of optimized computational tools.

II. METHODOLOGICAL BACKGROUND

This section provides background information to support later discussions. It is not intended as an exhaustive theoretical coverage of the subject matter.

A. Self-organizing map

The self-organizing map (SOM)[17] is an unsupervised neural network designed for dimensionality reduction while preserving topological relationships within data. SOMs consist of a grid of neurons, each associated with a weight vector representing a position in the input space. During training, the algorithm iteratively adjusts these weights to cluster similar data points close together on the map.

Two main training approaches exist: online and batch. Online training updates neuron weights iteratively as the model is fitted to the data. Since each calculation depends on previous results, there are limited opportunities to improve the speed of this method. Batch training, on the other hand, updates neuron weights by considering the entire dataset simultaneously. This method is primarily used in our data analysis and presents the greatest opportunity for computational speed-up, as it is inherently parallelizable.

The basic steps of SOM training are:

1. The weight vectors are initialized for all neurons.
2. Distances are calculated from each data point to all neurons.
3. The neuron closest to the current data point is identified, referred to as the best matching unit (BMU).
4. Weights are updated based on the BMU, its neighbors, and the current data point.

Steps 2–4 are then repeated until a prescribed number of occurrences (epochs) have been completed.[17] A more detailed explanation is provided in our previous work.[19,25]

B. Floating point numbers

Numerical representation and arithmetic in computers are at times difficult problems. While there are many differing ways to address these tasks, one of the most common approaches is the use of floating-point numbers. Floating-point numbers are used by computers to represent a large range of numbers within a fixed finite size, known as *precision*. This representation is a form of scientific notation in which the term *floating* refers to the ability of the decimal point to move, as determined by the exponent bits.

1. Precision

The IEEE 754 standard[26] defines various precisions for floating-point arithmetic. For our purposes, we focus on *single* (32-bit, fp32) precision and *double* (64-bit, fp64) precision. As an example, single precision is represented in Fig. 1. While MATLAB defaults to using double precision,[27] single precision can be implemented trivially with minimal code changes.

The importance of precision for our purposes is that using lower precision allows for more calculations within a given time on the same processor. Random-access memory (RAM) requirements are similarly reduced. The technical factors underpinning this benefit differ between CPUs and GPUs but both benefit from

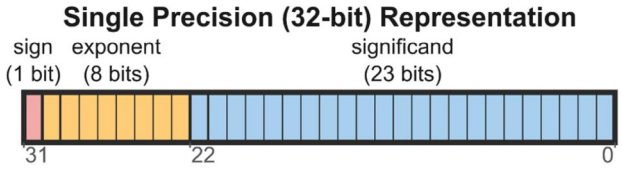

FIG. 1. Schematic representation of a 32-bit single-precision floating-point number in IEEE 754 (Ref. 26) format. The value comprises three components: a sign bit (1 bit), an exponent field (8 bits), and a significand (23 bits). Higher precisions (e.g., 64-bit double) use the same structure with different bit allocations.

the use of reduced precision. A related consideration is whether the accuracy and overall quality of the calculation can be retained with reduced precision, i.e., is higher precision essential to arrive at an acceptable answer?[28,29]

For CPUs, each core has a hardware component for floating point mathematics, called a floating-point unit (FPU) with a fixed *width*. As an example, a CPU with a 512-bit wide FPU, would be able to perform 8 double precision or 16 single precision calculations per core simultaneously.[30] Switching to single precision can, therefore, effectively double throughput. In contrast, many current and recent generation GPUs are optimized for single precision mathematics.[28,29] Performing double precision calculations on these GPUs may result in significant throughput penalties, with many GPUs being slower than CPUs in double precision calculations.

2. Arithmetic

Floating-point numbers are widely used as they provide a good compromise between speed and precision. However, representing a large range of numbers in a fixed amount of memory leads to inherent limitations.

One such limitation is that they are unable to represent every possible number. Each floating-point format has defined upper and lower bounds, additionally gaps exist between representable values within those bounds. An example of this constraint can be seen in irrational numbers, such as π, which cannot be precisely represented. However, this limitation rarely poses an issue. For example, in single precision, there are still 23 bits to represent the significand; for π, this provides more than sufficient precision for most calculations.

Arithmetic also poses limitations for floating-point numbers, which has the potential to impact large scale calculations, such as those in ML. The most prominent is the accumulation of rounding errors over many calculations. As a result, important mathematical properties can be violated, in particular, the associative property, $(a + b) + c = a + (b + c)$, and distributive property, $a(b + c) = ab + ac$. IEEE 754[26] makes every attempt to preserve these properties, but it cannot be guaranteed. This can become important in sensitive calculations such as eigenvector computation, where numerical stability must be carefully considered. A deeper discussion can be found in Higham's work[31] on numerical stability in algorithms, which provides a rigorous treatment of how rounding errors propagate through iterative numerical methods, directly relevant to the repeated matrix operations in SOM training.

C. Graphics processing units

GPUs have significantly more *cores* than CPUs.[32] The GPU cores are also sometimes called *threads* and will be referred to as such to differentiate them from CPU cores, which are architecturally distinct. Each of these threads contains a floating-point unit (FPU) and the thread can be thought of as equivalent to a single lane of the CPU's vector execution unit.

NVIDIA GPUs organize threads into groups of 32 called *warps*. These warps operate under a Single-Instruction Multiple-Thread (SIMT) model, similar in behavior to Single-Instruction Multiple-Data (SIMD), but with a few key differences.[32]

In SIMT, all threads in a warp execute the same instruction simultaneously. If threads follow different execution paths, as occurs in an if-else condition, the warp serializes those paths: executing one while the others remain idle, it then switches to the next path while the others remain idle, until all paths are executed. This divergence of threads introduces throughput penalties due to the reduction in parallel efficiency. Thus, peak efficiency occurs when all 32 threads of a warp agree on a single execution path.[32]

Warps themselves are grouped into Streaming Multiprocessors (SMs), which contain schedulers and caches; each SM functions like a CPU core. The SM allows for different warps to execute different instructions independently. However, the threads within a single warp must follow the same instruction path.[32]

Given that the SOM primarily involves matrix and vector dot products, it is trivial to divide these tasks into smaller chunks. This highly parallelizable nature makes it advantageous to perform these calculations on a GPU,[29,32] as we demonstrate in a later section.

III. EXPERIMENTAL AND METHODOLOGY

A. SOM-RPM toolbox enhancements

Considering the concepts discussed in Sec. II, several carefully selected enhancements were made to the SOM-RPM Toolbox to improve its computational runtime and efficiency. These key changes include:

Precision Selection: Allows the user to select single precision for calculations when appropriate to enhance computational efficiency and throughput. This new option lets users choose between single or double precision, enabling the user to find the right balance of precision and speed for their needs. Single precision offers improved throughput and reduced memory usage if extra precision is not necessary, and it also allows for better GPU resource utilization when available.

GPU Acceleration: To enhance the GPU's throughput, improvements were made to its implementation. Additionally, the code was modified to ensure that all relevant variables remain in GPU memory throughout training, minimizing costly data transfers between system RAM and GPU memory.

TABLE I. System configurations used for benchmarking. CPU core counts, frequencies, system RAM, GPU models, VRAM, architecture, and advertised floating-point performance (TFLOPS) for each system. N/A indicates no published specification.

Machine configurations for benchmarking			
	Desktop	Workstation	HPC
CPU	Intel Core i7-12700	Intel Xeon W-1390P	Intel Xeon Gold 6240R
Cores	12	8	96
Max frequency	4.90 GHz	5.30 GHz	4.0 GHz
System RAM	64GB	128GB	369GB
GPU	Nvidia T1000 8GB	Nvidia Quadro RTX 5000	Nvidia A100 PCI-e
VRAM	8 GB	16 GB	40 GB
GPU architecture	Turing	Turing	Ampere
FP64 TFlops	N/A	N/A	9.7
FP32 TFlops	2.5	11.2	19.5

Algorithmic Refinements: Various functions were optimized to eliminate redundant operations and improve parallelization, speeding up the revised functions.

These changes were designed to maintain model accuracy while significantly reducing execution time. A detailed description of the implementation changes can be found in the supplementary material (SI) (S.3–S.4).

B. Benchmarking approach

Overall execution time was assessed by comparing the epoch execution time of the original toolbox ($v1$) with the optimized version ($v2$) across multiple system configurations and data set sizes, detailed later in this section.

To obtain execution time, data sets were modelled using 8×8 neuron SOMs trained for 500 epochs. System time was recorded immediately before and after each epoch loop and then divided by the 500 epochs to calculate computation time per epoch. This was repeated five times for each configuration and combination of settings. The means of these repeats are reported, with a standard deviation cutoff of $\pm 3\sigma$ applied (affecting only one measurement). Initialization was excluded as it had a negligible impact on overall runtimes.

1. System configurations

Three system configurations were chosen to represent a typical range of computing capabilities that a researcher might have access to:

Desktop: Intel Core i7-12700 CPU, Nvidia T1000 GPU.
Workstation: Intel Xeon W-1390P CPU, Nvidia Quadro RTX 5000 GPU.
HPC: Intel Xeon Gold 6240R CPU, Nvidia A100 GPU.

Detailed specifications, including core counts, clock speeds, RAM, and GPU architecture, are summarized in Table I.

2. Dataset sizes

Three data sets were chosen to encompass a range of training data sizes that can be utilized in the SOM-RPM Toolbox, thereby encompassing the associated computational demands:

Small: Synthetic data set representative of a single hyperspectral ToF-SIMS map acquisition (IONTOF TOF.SIMS 5).
Medium: Case study[33] data set from a printed ink pattern used in the initial release of the SOM-RPM Toolbox.[25]
Large: Synthetic data set corresponding to MATLAB's GPU array memory limit.[27]

Detailed information on each data set, including dimensions and memory requirements for single and double precision is shown in Table II.

C. Execution time metrics

Toolbox execution time was assessed using average time per epoch as the primary metric, as outlined in the Benchmarking Approach. This metric was chosen because it directly reflects

TABLE II. Dataset dimensions and memory requirements for benchmarking. The Small dataset represents a typical single ToF-SIMS scan, Medium is the case study data from, (Ref. 33) and Large represents the maximum GPU array size in MATLAB (R2025a).

Data sizes for benchmarking			
	Standard scan "Small"	Previous case study "Medium"	Int32 Max "Large"
Dataset size	$512 \times 512 \times 1024$	$960 \times 800 \times 963$	intmax("int32")
Total entries	2.68E + 08	7.40E + 08	2.15E + 09
Double size (Gb)	0.27	0.74	2.15
Single size (Gb)	0.13	0.37	1.07

computational efficiency during SOM training, which is critical for real-time workflows.

The times for each run were placed into a pivot table to allow for comparisons between toolbox versions, system configurations and processor type (CPU vs GPU), and precision settings. An Excel file containing raw data and the pivot table is included in the supplementary material (SM), allowing the reader the freedom to make comparisons not covered in this paper.

The results section presents several key comparisons: overall runtimes across all configurations, the runtime impact of single vs double precision calculations, a benchmarking of GPU enhancements against the previous version, and the specific effects of software optimizations on execution time.

In these comparisons, we compared execution time for the *v*1 software to that of the *v*2 software to quantify the speed-up. Specifically, Speed-up = (*v*1 execution time)/(*v*2 execution time).

In addition, to evaluate whether reduced precision compromised model quality, we trained SOMs on the medium dataset using both single and double precision. Detailed calculations and data are provided in the supplementary material (S5).

D. Case study

A large, low spatial resolution ToF-SIMS image was collected on an unknown thin section of a rock sample, covering an analysis area of $4000 \times 6000\,\mu$m (800×1200 pixels). A 12×12 neuron SOM was selected to produce a computationally demanding configuration, providing a meaningful stress test of the toolbox improvements under realistic large-dataset conditions and was trained for 2000 epochs with each toolbox (*v*1 and *v*2) using the *workstation* to compare real world execution time.

1. Tof-SIMS experimental method

a. Data acquisition. A polished section of metamorphic rock, mounted in thin section form on a glass slide, was analyzed using ToF-SIMS hyperspectral mapping. The section contained multiple mineral phases including magnetite, fluorite, amphibole, and sulfide (pyrite), subsequently confirmed by an expert geologist using correlated transmitted and reflected light microscopy following ToF-SIMS data acquisition and SOM-RPM analysis. ToF-SIMS data were acquired with a TOF.SIMS 5 instrument (IONTOF GmbH, Germany) at room temperature under an ultrahigh vacuum. Samples were mounted using a TOF.SIMS 5 back-mount sample holder. A wide area scan was acquired ($4000 \times 6000\,\mu$m^2). A delayed extraction (DEX) scan was acquired from a smaller region of interest (ROI) within the larger scan.

ToF-SIMS images were captured using a 30 keV Bi_1^+ primary ion on an IONTOF V (IONTOF GmbH, Münster, Germany). Positive primary ion mode was used to capture $4000 \times 6000\,\mu$m^2 images using 800×1200 pixels, with $200 \times 200\,\mu$m^2 image patches. The images were rastered in a random mode, with a $100\,\mu$s cycle time capturing one shot per frame, and fifteen frames per patch. An electron flood gun, argon back fill to 1×10^{-6} mbar and a surface potential set to -40 V were used to reduce sample charging.

DEX images were captured using a 30 keV Bi_3^+ primary ion on the same instrument. Positive primary ion mode was used to capture data from the identified ROI, using thirty frames per patch. An electron flood gun, argon back fill, and a surface potential set to -52 V were used to reduce sample charging.

Spatially correlated optical images were acquired from the thin section for comparison using an Olympus SZX16-ILLTQ microscope.

b. Data preprocessing. Mass spectra collected from the thin section were manually interrogated. A characteristic peak list comprising 217 peaks each with greater than 750 counts was formed. Sample topology (differences in height) resulted in some peak splitting. These were eliminated from consideration. The peak selected data set was saved in the bif6 file format and imported into the MATLAB toolbox. A 12×12 neuron similarity map was trained for 2000 epochs on the Workstation with GPU acceleration enabled using the improved toolbox.

ROIs were then selected from the similarity map and returned to the ToF-SIMS instrument. Fast imaging (FI) and DEX scans were performed at higher resolution using the ROI.

IV. RESULTS AND DISCUSSION

Optimization of the SOM-RPM toolbox delivered substantial improvements in computational speed, achieving up to two orders of magnitude speed-up compared to the original implementation. While numerous comparisons can be made, we chose to highlight a select few in the following section aligning with the changes made to the toolbox. We start with the overall execution times for all configurations, exploring the speed-ups enabled by training in reduced precision, evaluating the improvement afforded by more efficient use of the GPU, and calculating execution time improvements associated with the three longest running functions. Finally, we report a case study which shows how these gains enable near real-time analysis of large ToF-SIMS data sets, a critical step toward integrating ML into live instrument workflows.

A. Overall execution time

Across all system configurations, the optimized toolbox (*v*2) consistently outperformed the original toolbox (*v*1). We present the average run time per epoch for each system configuration (Fig. 2), noting that, while the number of epochs varies by application, production models typically require several thousand to train a model.

There are four noteworthy observations from Fig. 2:

Overall Execution Time: The *v*2 toolbox is faster than the *v*1 toolbox in all tested configurations (Fig. 2).

Precision and Execution Time: As expected, for the *v*2 CPU, single precision takes approximately half the time of double precision (Fig. 3). Additionally, the *v*2 GPU sees even greater gains from going to single precision (Fig. 3), except for the HPC. This is also expected as most GPUs are optimized for single precision throughput.[28,29]

Improved Memory Efficiency: We can run the *v*2 GPU in more cases than we could on the *v*1. We see this with the medium data set on the desktop and the large dataset on the HPC

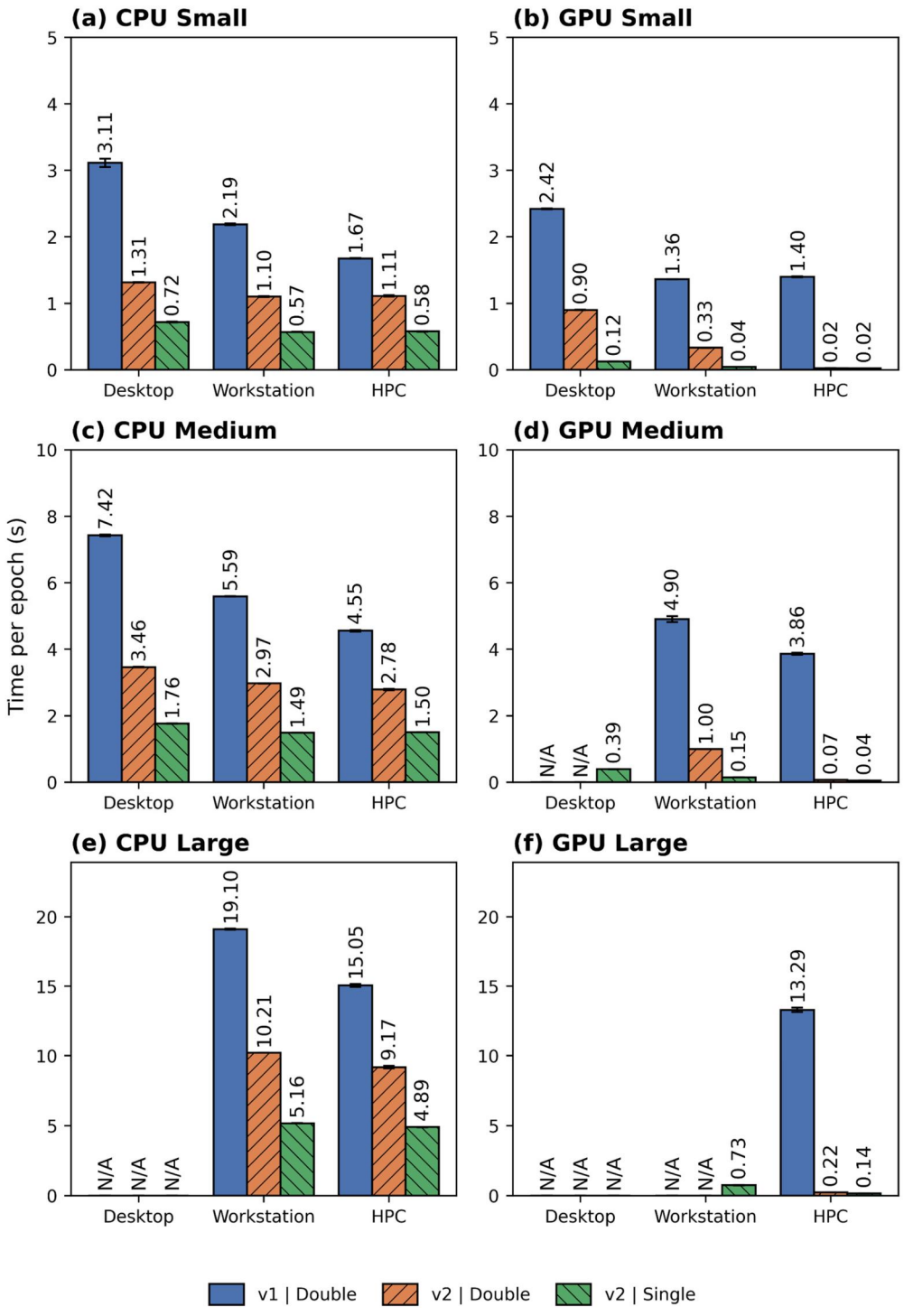

FIG. 2. Average time per epoch (s/epoch) for SOM-RPM training. Panels show Small [(a) and (b)], Medium [(c) and (d)], and Large [(e) and (f)] datasets on CPU [(a), (c) and (e)] and GPU [(b), (d) and (f)]. Bars represent original toolbox (v1, double precision) and optimized toolbox (v2, both precisions) across three system configurations: Desktop (Intel i7-12700/Nvidia T1000), Workstation (Xeon W-1390P/Quadro RTX 5000), and HPC (Xeon Gold 6240R/A100). N/A indicates insufficient memory to train on that configuration.

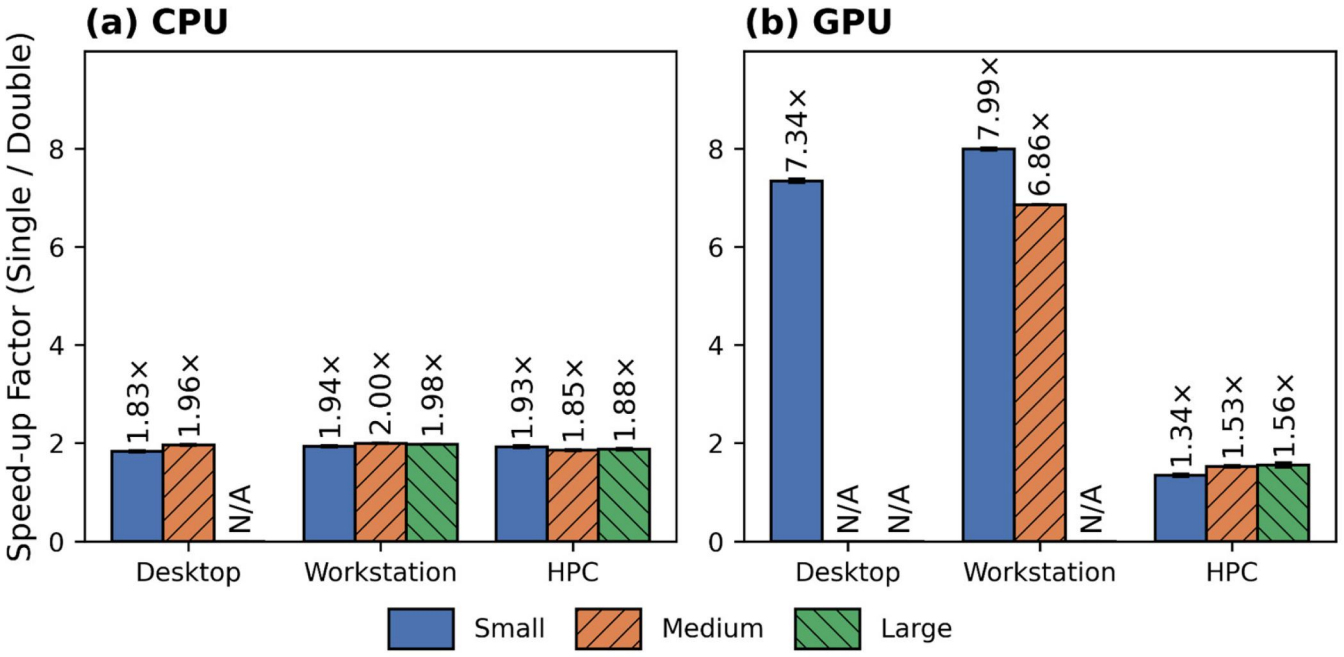

FIG. 3. Speed-up from switching to single precision, calculated as (double precision time)/(single precision time). (a) CPU and (b) GPU results across three system configurations and dataset sizes. A ratio of 2.0 indicates single precision runs in half the time. N/A indicates insufficient memory to train on that configuration.

(Fig. 2). This shows the improved memory efficiency of the *v*2 toolbox.

Processor Efficiency: Notably, the optimized CPU implementation in *v*2 outperformed the *v*1 GPU code (Fig. 5), highlighting the critical importance of efficient processor utilization and memory management.

B. Precision and accuracy

A critical consideration in runtime optimization is whether reduced numerical precision compromises the quality of the ML model. As noted in the methodology, ToF-SIMS data is inherently noisy,[1,2] suggesting that the precision of 64-bit arithmetic may be unnecessary for reliable spectral interpretation.

1. Execution speed

Figure 3 shows the speed improvement by moving from double precision to single precision calculations. Increases for the CPU [Fig. 3(a)] were consistently near the theoretical max of 2×, while even greater gains can be seen on the GPU [Fig. 3(b)]. A notable exception was a selection of calculations performed on the HPC, as it used a GPU optimized for double precision.

2. Numerical difference and suitability for ToF-SIMS workflows

To evaluate whether reduced precision compromised model quality, two SOMs were trained on the medium data set under identical conditions, differing only in floating-point precision, one in double (64-bit) and one in single (32-bit). The resulting weight matrices were compared element-wise by computing a difference matrix; full details of the methodology and statistical analysis are provided in the supplementary material (S5–S6).

We found the numerical difference to be negligible. The mean absolute difference between individual entries in the weight matrices was less than 1×10^{-5}, and both precision settings produced weight matrices with an identical standard deviation of 0.0723, indicating the overall structure and distribution of the learned representations were preserved despite the reduced bit depth.

These results confirm that single precision was sufficient for these ToF-SIMS workflows. The accumulation of rounding errors associated with single precision remained several orders of magnitude below the inherent signal-to-noise ratio of the mass spectral acquisitions. Consequently, the use of single precision represents a highly favorable trade-off, enabling near-real-time analysis without sacrificing the reliability of chemical identification.

C. GPU improvements

The most significant runtime gains associated with *v*2 resulted from restructured GPU utilization, specifically targeting the limitations of the *v*1 implementation. In the original codebase, the GPU was frequently bottlenecked by the requirement to transfer data between system RAM and video random-access memory (VRAM) during active training epochs. By ensuring that all relevant variables

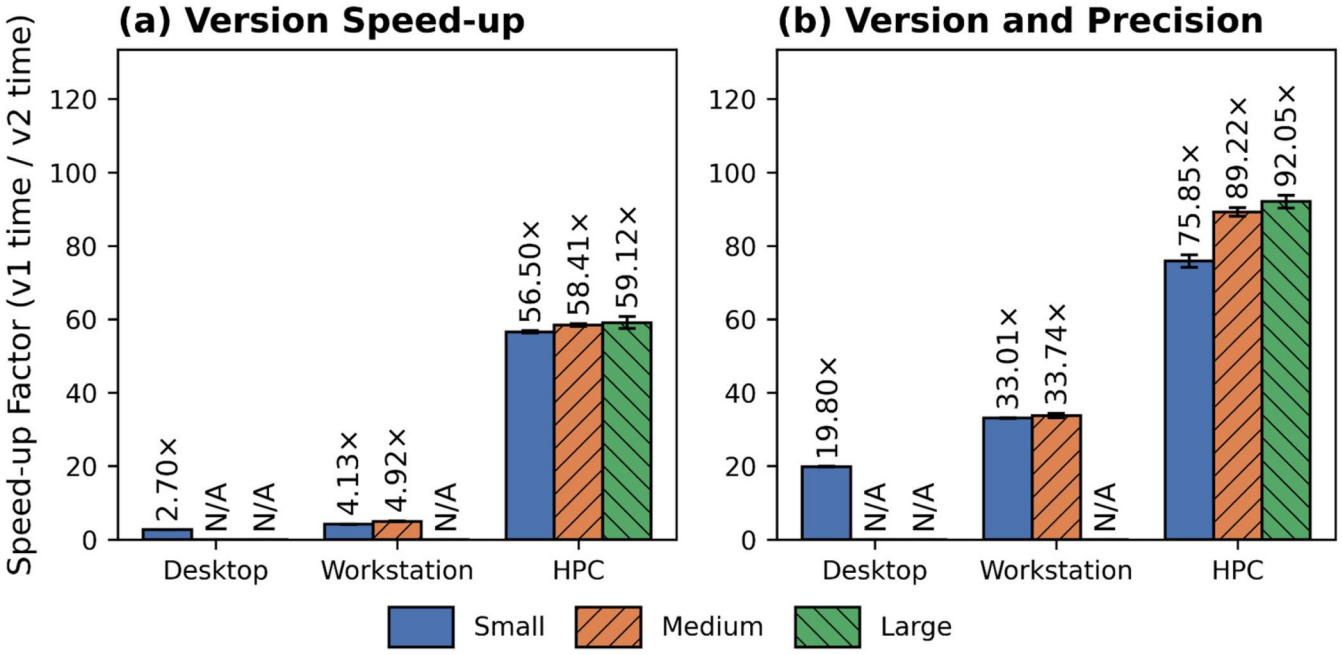

FIG. 4. GPU speed-up from v1 to v2. (a) Both versions in double precision, isolating algorithmic improvements. (b) v1 double precision vs v2 single precision, showing combined gains. Speed-up = (v1 time)/(v2 time) per epoch. N/A indicates insufficient memory to train on that configuration.

remain in VRAM throughout the entire duration of the SOM training, *v*2 effectively eliminates this I/O-related latency.

Furthermore, the introduction of single-precision arithmetic aligns the software with the native architecture of GPUs, which are typically optimized for single precision floating-point throughput.[28,29] The impact of these combined processor-level enhancements is detailed in Fig. 4, which illustrates the speed-up factors achieved when comparing the *v*2 GPU-optimized implementation against the previous GPU execution paths.

We see the improvements in the changes to how the toolbox makes use of the GPU in Fig. 4. Comparing GPU throughput in identical, double, precision [Fig. 4(a)] we see a speed-up across all system configurations, showing the improvements in the efficiency of how the toolbox utilizes a GPU. Further showcasing these improvements are the gains in the HPC. As the HPC is the only system configuration tested that has optimizations for GPU double precision, we get to see a clear picture of the improved processor efficiency of the *v*2 toolbox. Additionally, we can see the improvements in using single precision on the GPU on the v2 toolbox [Fig. 4(b)]. The Desktop and Workstation GPUs represent the majority of GPUs, which are optimized for single precision throughput,[28,29] and we subsequently see large speed-ups by using single precision on them, whereas the HPC sees a speed-up that is in line with what would be expected from halving precision.

D. Software improvements

The transition from *v*1 to *v*2 was underpinned by targeted algorithmic refinements that addressed specific bottlenecks in the SOM training loop. The importance of these software improvements can be seen in Fig. 5.

Using the CPU as an example, we can see that in a like-for-like comparison there are improvements across the board for all system configurations and model sizes [Fig. 5(a)]. But to further highlight the effect of the improvements, the *v*2 CPU outperforms the *v*1 GPU in identical precision [Fig. 5(b)]. Underscoring that efficient software design and proper memory management are more critical to runtime than the availability of more powerful systems.

To further quantify the improvements, we look at three major components that occupied most of the training time:

Distance Computation: This calculates the Euclidean distance between the weight and data vectors.
Updating Weights: Moves the winning neuron and its topological neighbors closer to the data vector.
Quantization Error Calculation: Calculates the average between each input data point and the winning neuron to measure how well the neuron weight vectors represent the data.

Table III shows the components and the time they took to run with the old code, and the new code (GPU acceleration on and off) and the resulting speed up of the new code (GPU acceleration on and off). This is visualized in Fig. 6.

For the first two software changes, Distance Calculation and Updating Weights, we see that there are large improvements in efficiency across all system configurations (Fig. 6). However, the Quantization Error only improves on the GPU. This is

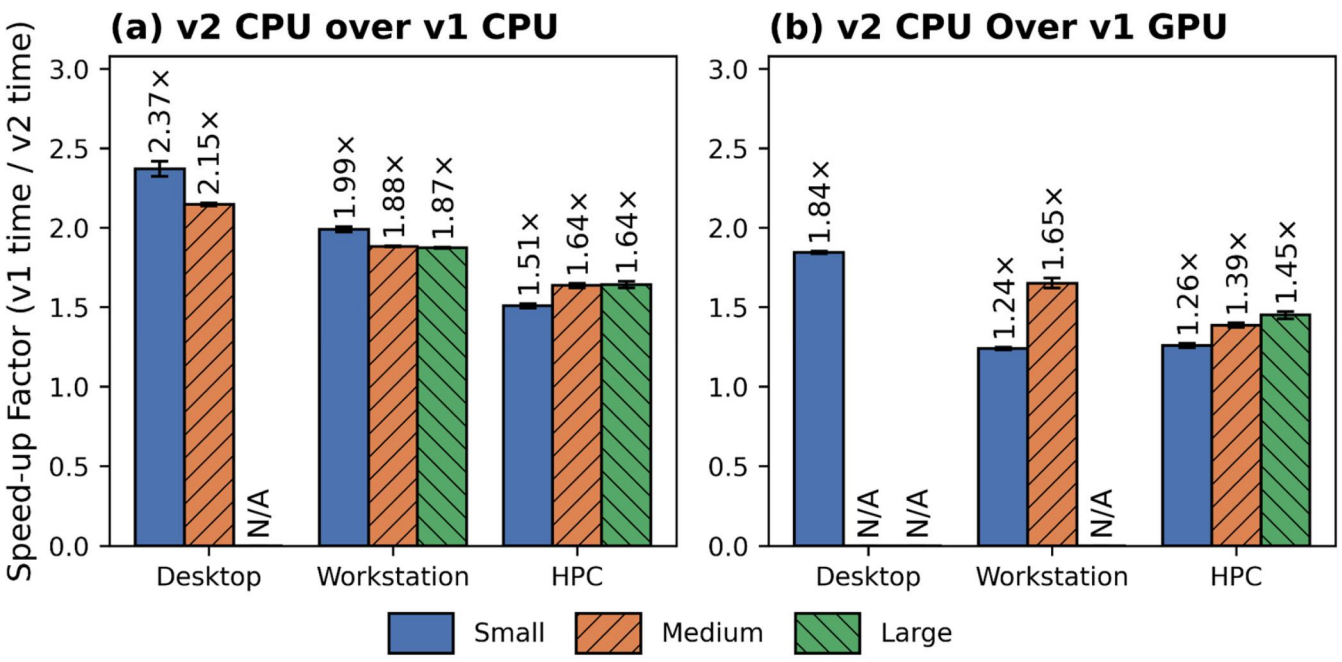

FIG. 5. Computational speed-up (old/new computation time per epoch) comparison of software improvements. Comparing *v*1 CPU vs *v*2 CPU with identical precision (a), and *v*1 GPU vs *v*2 CPU with identical precision (b). Panel (b) demonstrates that optimized software on CPU can outperform unoptimized GPU code. Larger ratios indicate greater speed-up. N/A indicates insufficient memory to train on that configuration.

because the GPU benefits from batching whereas the CPU does not.

One of the benefits of batching is it allows for better use of VRAM on the GPU. If a batch size is too large it can result in the GPU running out of memory, we see this in the *v*2 toolbox where the Desktop [Fig. 2(d)] and Workstation [Fig. 2(f)] are able to fit models to larger data sets on the same GPU.

Additionally, batching can provide runtime improvement even for data that can fit on the GPU. Figure 7 shows a representative case of execution time versus number of batches. The pattern of these data held consistent across our testing, specifically that there is an ideal batch size (number of batches), dependent on various factors, and execution time increases the further away from that ideal size. Critically the execution time does not increase evenly. With too few batches having a far more severe penalty than too many (Fig. 7).

E. Case study

A key application of this speed improvement is that it enables real-time analysis of ToF-SIMS data, via a survey-then-target workflow. With this approach, a wide area scan is first acquired at low spatial resolution to capture the chemical landscape of the sample. The SOM-RPM toolbox is then used to analyze the data set, generating a similarity map in which each pixel is color coded according to its dominant SOM neuron, encoding spectral relationships across the entire survey. Potential ROIs can identified from this survey, for example, as spatially localized regions

TABLE III. Execution time (seconds) for three SOM training operations on the medium dataset (Workstation), comparing v1 CPU, v2 CPU (double precision), and v2 GPU (single precision). Speed-up calculated as v1 time/v2 time.

	Software improvements				
Operation	V1 CPU run time (s)	V2 CPU run time (s)	V2 GPU run time (s)	V1 CPU to V2 CPU speed-up (V1/V2)	V1 CPU to V2 GPU speed-up (V1/V2)
Distance calculation	2580.31	740.49	83.46	3.48	30.92
Updating weights	3215.01	1202.30	48.70	2.67	66.02
Quantization error calculation	2533.84	2547.94	143.33	0.99	17.68

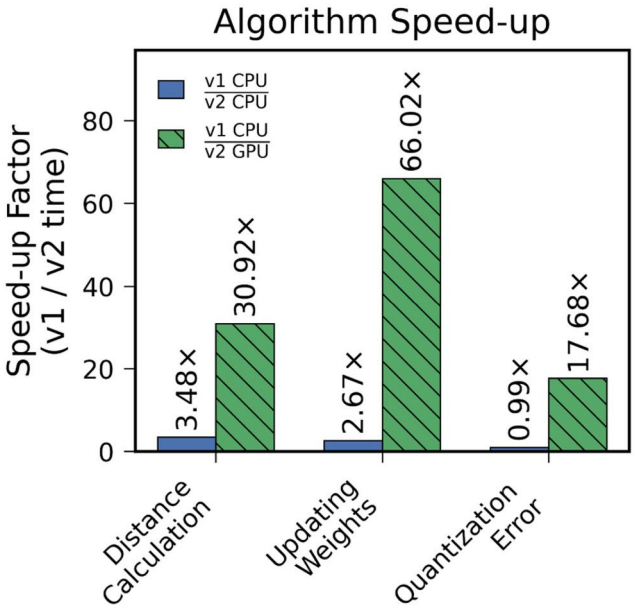

FIG. 6. Computational speed-up from *v1* to *v2* for three core SOM training functions: distance calculation, weight update, and quantization error. Blue bars show CPU-only speed-up (*v1* CPU/*v2* CPU, double precision); green hatched bars show combined CPU-to-GPU speed-up (*v1* CPU/*v2* GPU, single precision). Speed-up values are labeled above each bar. All measurements from the medium dataset on the Workstation configuration.

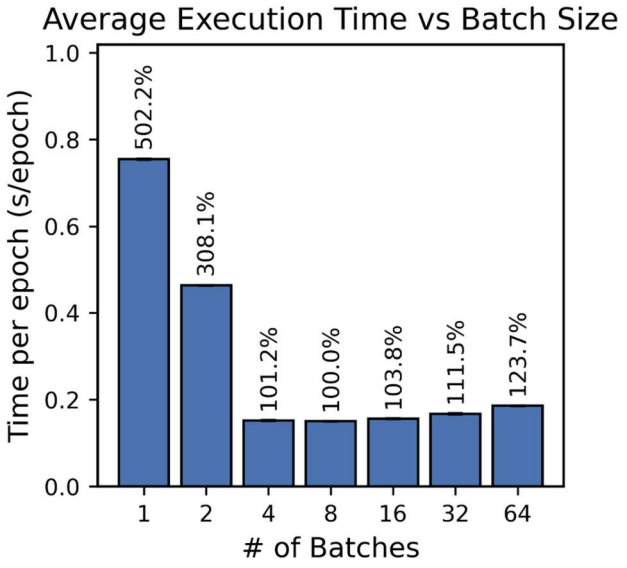

FIG. 7. Average execution time per epoch (s/epoch) as a function of batch count for the medium dataset on the HPC GPU (Nvidia A100, single precision). Percentage values indicate execution time relative to the optimal batch count. The minimum occurs at 8 batches; execution time increases asymmetrically, with fewer batches imposing a steeper penalty than excess batches.

exhibiting chemical heterogeneity (evidenced by a diversity of neuron color assignments), or unique chemical compositions (i.e., characterized by a unique color composition that differentiates the region from surrounding material). For mineral samples, this process can identify the presence of spectrally anomalous mineral phases not easily discernible from the high-dimensional data. Critically, as SOM-RPM is unsupervised, performing this data driven selection process requires no prior compositional knowledge of the sample.

Following the ROI selection process, the next step in the workflow is to perform a high spatial resolution scan targeting the ROI, which allows for chemical composition to be resolved at submicron lateral resolution, as enabled by ToF-SIMS.[34] This lateral resolution enables detailed mapping of elemental and molecular distributions within selected surface regions,[1] at scales that capture fine-grained spatial features; high lateral resolution ToF-SIMS has been used extensively for geological samples in particular,[34] enabling accurate and surface-sensitive mapping of trace elements and mineral phases in rock sections.[35,36]

In previous work, we applied a similar survey-then-target workflow for biological tissue analysis, where whole section ToF-SIMS survey data were used to generate machine learning derived attention maps, from which ROIs were selected for high spatial resolution reimaging to resolve molecular features at subcellular scale.[37] We note that the current work differs in that it utilizes an entirely unsupervised workflow (i.e., SOM-RPM).

The practicality of this survey-then-target workflow depends critically on the SOM-RPM analysis completing within the timeframe of the wide-area acquisition, so that ROI selection and instrument redirection can occur without workflow interruption. The computational speed improvements of the v2 toolbox directly enabled this. To demonstrate this, we performed a wide-area ToF-SIMS scan of a polished thin section of metamorphic rock containing a chemically diverse assemblage of mineral phases, with results presented in Fig. 8. This sample was selected as a case study for several reasons: First, the sample exhibits pronounced chemical heterogeneity across distinct mineral phases, which can be independently verified by optical microscopy under transmitted and reflected light [Fig. 8(a)]. This provides an instrument independent ground truth against which the SOM-RPM outputs can be evaluated, without relying solely on the mass spectral data itself. Second, the wide area scan ($4000 \times 6000\,\mu$m at 800×1200 pixels, 217 spectral channels) provides a data set of sufficient scale to meaningfully stress-test the runtime improvements. Third, the analytical challenge posed by a spatially heterogeneous, multiphase inorganic sample is structurally representative of the broader diversity of ToF-SIMS use cases to which the SOM-RPM toolbox has been applied, including biological tissue,[37] biomaterial surfaces,[5] and polymeric materials.[38,39] Moreover, previous applications of ToF-SIMS for geological samples have demonstrated its effectiveness for mineral identification and trace element mapping in rock sections.[35,36] The complex metamorphic rock, therefore, provides a generally informative demonstration exemplar.

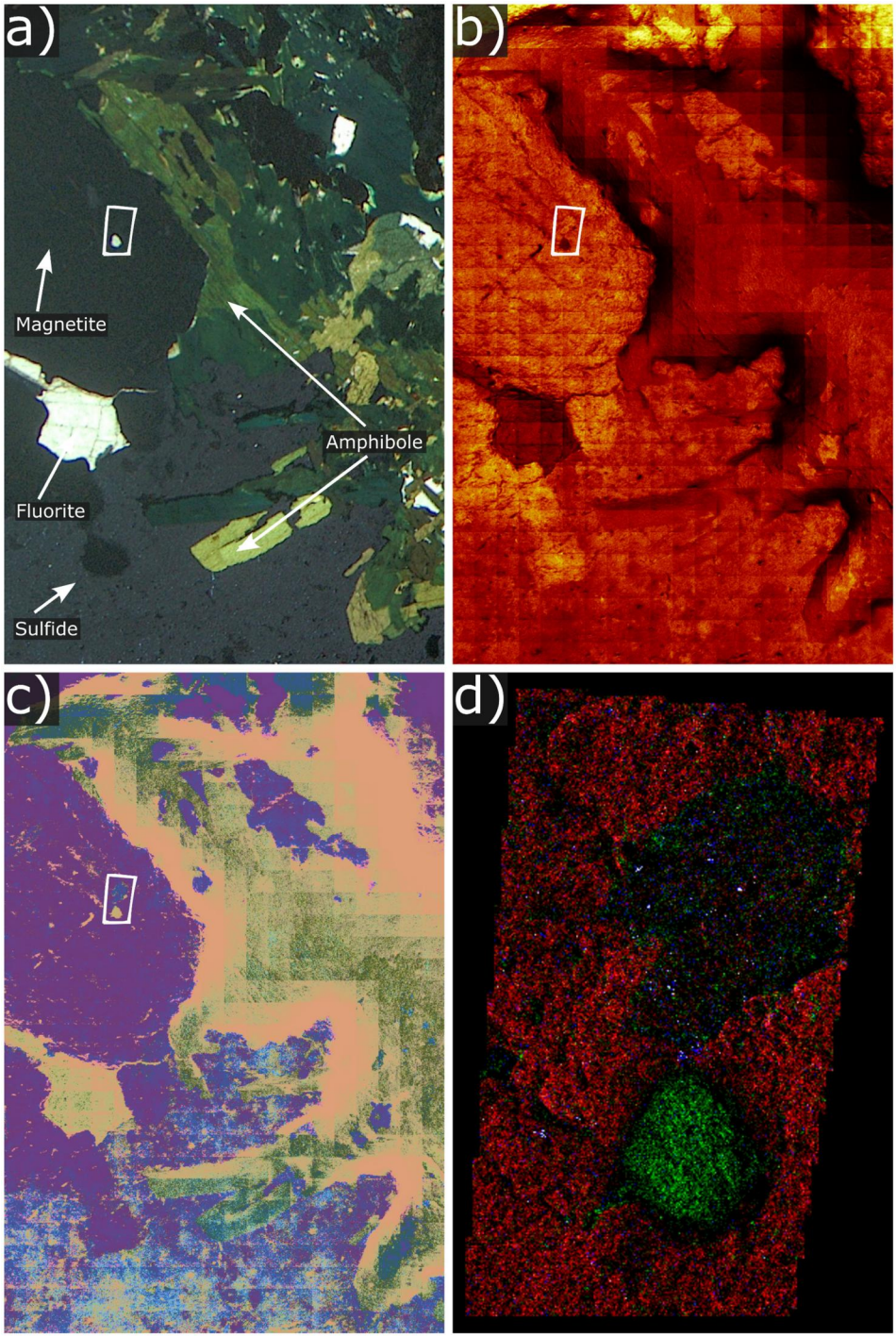

FIG. 8. Multi-modal characterization of a metamorphic rock thin section. (a) Optical microscope image (transmitted and reflected light) with mineral phases labeled by expert petrographic analysis. (b) Total ion image from wide-area ToF-SIMS mapping ($4000 \times 6000\,\mu$m, 800×1200 pixels, 30 keV Bi_1^+). The white box indicates the region of interest (ROI) selected for high-resolution analysis. (c) SOM-RPM similarity map (12×12 neuron SOM, 2000 epochs) of the same area; white box indicates the region of interest (ROI) selected for high-resolution analysis. (d) RGB elemental composite of the ROI acquired in delayed extraction (DEX) mode using 30 keV Bi_3^+: Fe^+ (red), Ca^+ (green), Al^+ (blue).

This data set, with its well-defined phase boundaries and independent optical ground truth, provides a realistic test of the toolbox under conditions representative of complex, multicomponent ToF-SIMS analyses. Using the *v*2 toolbox, we produced a SOM-RPM similarity map [Fig. 8(c)] using the acquired data. Figure 9 shows the mass spectra extracted from each identified region, as marked in Fig. 8(a).

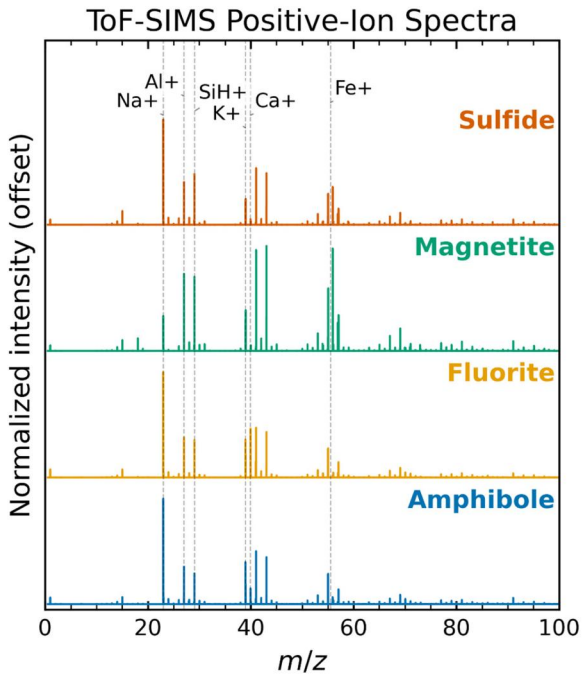

FIG. 9. Representative positive-ion ToF-SIMS mass spectra (m/z 0–100) extracted from four SOM-RPM-identified regions in Fig. 8(c), corresponding to sulfide (pyrite), magnetite, fluorite, and amphibole. Spectra are normalized and offset vertically for clarity. Dashed lines indicate selected characteristic peaks: Na^+, Al^+, SiH^+, K^+, Ca^+, and Fe^+.

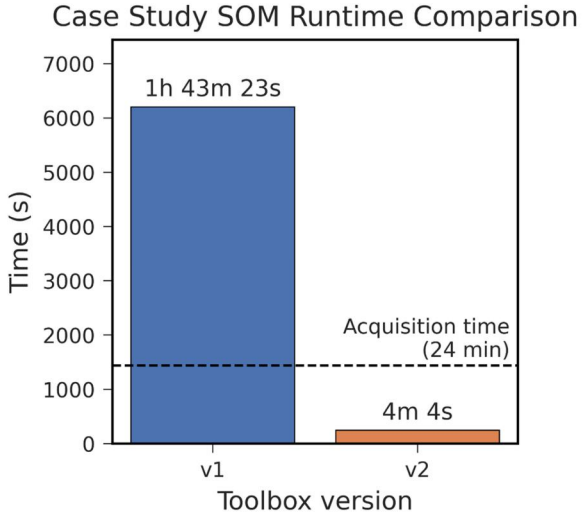

FIG. 10. Total processing time for the case study dataset (1200 × 800 × 217, 12 × 12 neuron SOM, 2000 epochs) using the v1 and v2 SOM-RPM toolboxes on the Workstation with GPU. The dashed line indicates the wide-area scan acquisition time (24 min), demonstrating that v2 processing is completed well within the instrument acquisition window, enabling genuine real-time analysis.

Using the *v*2 toolbox, total computation time was 4 min 4 s, using the workstation with GPU utilization. In comparison, total computation time using the *v*1 toolbox on the same system was 1 h 43 min 23 s, representing a speed-up factor of approximately 25× (Fig. 10). For context, acquisition of the wide-area scan required 24 min, meaning the v2 toolbox produces a completed similarity map well within the time the instrument remains staged and ready for follow-up acquisitions, a prerequisite for any genuinely real-time workflow. This is critical, as such a speed-up enables real-time decisions to be made live while using the instrument. Furthermore, it presents opportunities for more readily implementing automated or human-in-the-loop workflows, for example, utilizing an instrument application programming interface (API) to automate data analysis and provide a feedback loop for instrument control.[21–23] A final advantage is that SOMs usually benefit from hyperparameter optimization,[17,19,25] so having low computation time enables better exploration when doing optimization of the SOM architecture.

Using the SOM-RPM similarity map as a guide, we performed a high-lateral resolution scan focused on a small area exhibiting spectral heterogeneity and spatial regions of unique composition [as shown by the white box across Figs. 8(a)–8(c)]. Figure 8(d) presents the maps of spatially heterogeneous metallic ions Fe^+, Ca^+, and Al^+ within the ROI [Fig. 8(d)]. These maps reveal the discrete compositional phases within the ROI, at a greater level of detail than was possible with the low-resolution survey data alone.

Using this case study as an example, our similarity map allowed us to: (1) identify a meaningful and interesting spatial feature from the low spatial resolution data; (2) immediately create a ROI for subsequent high spatial resolution data acquisition; (3) identify the chemical composition of mineral grains that are not visible on the optical image [Fig. 8(a)] and barely visible within the total ion scan [Fig. 8(b)]. This showcases the practical value of GPU-optimization for this kind of real-time experimental workflow. These capabilities, previously impractical with the v1 toolbox, mark a transition from offline post-processing to a genuinely instrument-integrated analytical workflow.

V. SUMMARY AND CONCLUSIONS

In this study, we presented practical, targeted optimizations to the previously released SOM-RPM toolbox, demonstrating how targeted changes can dramatically improve computational efficiency. These enhancements, ranging from processor-aware precision adjustments to algorithmic refinements, enabled up to two orders of magnitude speed-up, significantly reducing latency in ToF-SIMS data analysis workflows.

Our benchmarking across diverse system configurations and data sizes confirms the robustness and scalability of these improvements. The case study further illustrates the practical utility of the enhanced toolbox in guiding high-resolution scans and accelerating research workflows.

Beyond the technical improvements, this work underscores the importance of computational efficiency in scientific software development and provides a resource for researchers developing optimized computational tools.

SUPPLEMENTARY MATERIAL

Supplementary material includes additional details on software changes, the case study methods, more details on the variation between models fitted in single and double precision, and an Excel sheet with the run times and a pivot table to allow researchers to filter the data and do their own comparisons of various metrics.

ACKNOWLEDGMENTS

B.O. acknowledges support provided through a La Trobe University Postgraduate Research Scholarship. The authors S.E.B., W.G., and P.J.P. acknowledge support by the Office of National Intelligence—National Intelligence Discovery Grants (Nos. NI210100127 and NI240100140) funded by the Australian Government. This work was performed in part at the Australian National Fabrication Facility (ANFF), a company established under the National Collaborative Research Infrastructure Strategy, through the La Trobe University Centre for Materials and Surface Science. The authors acknowledge the Milano Chemometrics and QSAR Research Group for the development of the Kohonen and CP-ANN Toolbox for MATLAB.[40,41]

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Brian Oslinker: Conceptualization (equal); Data curation (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Wil Gardner:** Conceptualization (equal); Methodology (equal); Software (equal); Writing – review & editing (equal). **Sarah E. Bamford:** Data curation (equal); Investigation (equal); Methodology (equal); Writing – original draft (equal); Writing – review & editing (equal). **See Yoong Wong:** Validation (equal); Writing – review & editing (equal). **John. A. Webb:** Investigation (equal); Writing – review & editing (equal). **Davide Ballabio:** Software (equal); Writing – review & editing (equal). **Thomas M. Kohl:** Writing – review & editing (equal). **Paul J. Pigram:** Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request. The "medium" data set used can be found in La Trobe OPAL at https://doi.org/10.26181/25648905, Ref. 33.

REFERENCES

[1]N. P. Lockyer, S. Aoyagi, J. S. Fletcher, I. S. Gilmore, P. A. W. van der Heide, K. L. Moore, B. J. Tyler, and L.-T. Weng, Nat. Rev. Methods Primers **4**, 32 (2024).
[2]J. C. Vickerman and D. Briggs, *TOF-SIMS: Materials Analysis by Mass Spectrometry*, 2nd ed. (SurfaceSpectra, Manchester, 2013).
[3]L. D. Gelb and A. V. Walker, J. Vac. Sci. Technol. B **36**, 03F (2018).
[4]N. Verbeeck, R. M. Caprioli, and R. van de Plas, Mass Spectrom. Rev. **39**, 245 (2020).
[5]W. Gardner, D. A. Winkler, B. W. Muir, and P. J. Pigram, Biointerphases **17**, 020802 (2022).
[6]I. T. Jolliffe, *Wiley StatsRef: Statistics Reference Online* (Wiley, Hoboken, NJ, 2014).
[7]I. T. Jolliffe and J. Cadima, Philos. Trans. R. Soc. A **374**, 20150202 (2016).
[8]S. Lloyd, IEEE Trans. Inf. Theory **28**, 129 (1982).
[9]J. MacQueen, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (University of California, Berkeley, 1967), p. 281.
[10]A. R. Konicek, J. Lefman, and C. Szakal, Analyst **137**, 3479 (2012).
[11]L. van der Maaten and G. Hinton, J. Mach. Learn. Res. **9**, 2579 (2008).
[12]J. M. Fonville *et al.*, Anal. Chem. **85**, 1415 (2013).
[13]T. Smets, N. Verbeeck, M. Claesen, A. Asperger, G. Griffioen, T. Tousseyn, W. Waelput, E. Waelkens, and B. De Moor, Anal. Chem. **91**, 5706 (2019).
[14]T. Smets, E. Waelkens, and B. De Moor, Anal. Chem. **92**, 5240 (2020).
[15]T. Smets, T. De Keyser, T. Tousseyn, E. Waelkens, and B. De Moor, Anal. Chem. **93**, 3452 (2021).
[16]P. Franceschi and R. Wehrens, Proteomics **14**, 853 (2014).
[17]T. Kohonen, Neural Netw. **37**, 52 (2013).
[18]X. Xiong *et al.*, J. Am. Soc. Mass Spectrom. **23**, 1147 (2012).
[19]S. E. Bamford, W. Gardner, D. A. Winkler, B. W. Muir, D. Alahakoon, and P. J. Pigram, J. Am. Soc. Mass Spectrom. **35**, 2516 (2024).
[20]K. Krijnen, P. Blenkinsopp, R. M. A. Heeren, and I. G. M. Anthony, J. Am. Soc. Mass Spectrom. **35**, 3063 (2024).
[21]S. Kandel *et al.*, Nat. Commun. **14**, 5501 (2023).
[22]S. V. Kalinin *et al.*, npj Comput. Mater. **9**, 227 (2023).
[23]S. R. Spurgeon *et al.*, Nat. Mater. **20**, 274 (2021).
[24]H. Hu, D. Helminiak, M. Yang, D. Unsihuay, R. T. Hilger, D. H. Ye, and J. Laskin, ACS Meas. Sci. Au **2**, 466 (2022).
[25]S. E. Bamford, W. Gardner, D. Ballabio, B. Oslinker, D. A. Winkler, B. W. Muir, and P. J. Pigram, Chemom. Intell. Lab. Syst. **261**, 105383 (2025).
[26]IEEE, *IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019* (IEEE, New York, 2019).
[27]The MathWorks Inc., *MATLAB Version 25.1 (R2025a)* (The MathWorks, Natick, MA, 2025).
[28]M. Rexroth, C. Schäfer, G. Fourestey, and J.-P. Kneib, Astron. Comput. **30**, 100340 (2020).
[29]P. Micikevicius *et al.*, arXiv:1710.03740 (2017).
[30]Intel Corporation, *Intel® 64 and IA-32 Architectures Software Developer's Manual* (Intel, Santa Clara, CA, 2024).
[31]N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. (Society for Industrial and Applied Mathematics, Philadelphia, 2002).
[32]NVIDIA Corporation, *CUDA C++ Programming Guide, Release 13.2* (NVIDIA, Santa Clara, CA, 2026).
[33]S. E. Bamford, W. Gardner, P. J. Pigram, B. W. Muir, D. A. Winkler, and D. Ballabio (2024). "Data set for a comprehensive tutorial on the SOM-RPM toolbox for MATLAB," La Trobe OPAL. https://doi.org/10.26181/25648905
[34]R. North, L. T. White, M. Nancarrow, A. Dosseto, and D. Tanner, Geostand. Geoanal. Res. **47**, 125 (2023).
[35]T. Long, S. W. J. Clement, H. Xie, and D. Liu, Int. J. Mass Spectrom. **450**, 116289 (2020).
[36]S. Rinnen, C. Stroth, A. Rißе, C. Ostertag-Henning, and H. F. Arlinghaus, Appl. Surf. Sci. **349**, 622 (2015).
[37]W. Gardner *et al.*, Anal. Chem. **98**, 6692 (2026).
[38]S. Y. Wong *et al.*, Adv. Mater. Interfaces **10**, 2202334 (2023).
[39]S. E. Bamford, D. Yalcin, W. Gardner, D. A. Winkler, T. M. Kohl, B. W. Muir, S. Howard, E. A. Bruton, and P. J. Pigram, Anal. Chem. **95**, 7968 (2023).
[40]D. Ballabio, V. Consonni, and R. Todeschini, Chemom. Intell. Lab. Syst. **98**, 115 (2009).
[41]D. Ballabio and M. Vasighi, Chemom. Intell. Lab. Syst. **118**, 24 (2012).