



OPEN Automated image segmentation for accelerated nanoparticle characterization

Alexandra L. Day¹, Carolin B. Wahl^{2,3}, Roberto dos Reis^{2,3,4}, Wei-keng Liao¹, Youjia Li¹, Muhammed Nur Talha Kilic⁵, Chad A. Mirkin^{2,3,6}, Vinayak P. Dravid^{2,3,4}, Alok Choudhary¹ & Ankit Agrawal¹✉

Recent developments in materials science have made it possible to synthesize millions of individual nanoparticles on a chip. However, many steps in the characterization process still require extensive human input. To address this challenge, we present an automated image processing pipeline that optimizes high-throughput nanoparticle characterization using intelligent image segmentation and coordinate generation. The proposed method can rapidly analyze each image and return optimized acquisition coordinates suitable for multiple analytical STEM techniques, including 4D-STEM, EELS, and EDS. The pipeline employs computer vision and unsupervised learning to remove the image background, segment the particle into areas of interest, and generate acquisition coordinates. This approach eliminates the need for uniform grid sampling, focusing data collection on regions of interest. We validated our approach using a diverse dataset of over 900 high-resolution grayscale nanoparticle images, achieving a 96.0% success rate based on expert-validated criteria. Using established 4D-STEM acquisition times as a baseline, our method demonstrates a 25.0 to 29.1-fold reduction in total processing time. By automating this crucial preprocessing step and optimizing data acquisition, our pipeline significantly accelerates materials characterization workflows while reducing unnecessary data collection.

Nanoparticle characterization is rapidly becoming a “big data” problem^{1–9}, where the volume of data threatens to surpass the capabilities of analytical approaches. Recent developments have made it possible to synthesize vast arrays with millions of distinct nanoparticles on a chip¹⁰, known as *combinatorial megalibraries*. Manual analysis can no longer fully capture the abundance of information contained in these arrays, leading to a bottleneck in processing. To meet the challenge posed by high throughput characterization, researchers are increasingly turning to automated tools and analysis frameworks. Recently, machine learning (ML), deep learning (DL), and artificial intelligence (AI) have become valuable tools in materials science^{11–20}. Researchers are compiling databases and libraries of material properties^{21,22} and developing analysis frameworks that leverage the power of AI^{23,24}, which are increasingly driving materials design and discovery^{25–28}.

This paper details the second phase of a project to automate key steps in nanoparticle characterization. The first step consists of a quality assessment to determine if a particle meets the criteria for further analysis. In previous work^{29–31}, we built binary classification machine learning models for this task. Once a particle has been selected for in-depth study, researchers must choose where within the particle to gather data. In this work, we automate this step with a custom pipeline that returns acquisition coordinates for STEM-based analysis, such as Four-Dimensional Scanning Transmission Electron Microscopy (4D-STEM)³², Energy Dispersive X-ray Spectroscopy (EDS)³³, and Electron Energy Loss Spectroscopy (EELS)³⁴. Our process incorporates techniques from computer vision and unsupervised learning and was carefully designed and tested with input from microscopy and materials science experts on our team. We are working to deploy these automated tools in the laboratory to accelerate 4D-STEM processing.

¹Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208, USA.

²Department of Materials Science and Engineering, Northwestern University, Evanston, IL 60208, USA.

³International Institute for Nanotechnology, Northwestern University, Evanston, IL 60208, USA. ⁴The NUANCE Center, Northwestern University, Evanston, IL 60208, USA. ⁵Department of Computer Science, Northwestern University, Evanston, IL 60208, USA. ⁶Department of Chemistry, Northwestern University, Evanston, IL 60208, USA. ✉email: ankit-agrawal@northwestern.edu

Challenges and objectives

Current characterization workflows face significant efficiency limitations. Standard approaches collect high-resolution information using an evenly spaced grid of 128×128 points, dividing each image into 16,384 smaller square regions for analysis. This grid-based approach presents several challenges: it produces regions of identical size regardless of feature complexity, fails to distinguish between particle and background areas, and leads to inefficient use of both time and data storage resources. An optimized preprocessing method should, therefore, accomplish three key objectives: (1) background identification and removal; (2) automated image segmentation based on key features; and (3) appropriate placement of acquisition boxes. Figure 1 illustrates this desired workflow.

For clarity, we define key terms used throughout this paper. We refer to the original microscopy image as the "parent image," and the smaller derivative images as "child images." The term "acquisition box" describes the set of square coordinates that defines each child image's position within a parent frame (Fig. 1).

We present a method that accelerates the characterization workflow by more than 25 times on average, validated using a unique dataset of 964 grayscale nanoparticle images. The pipeline is specifically designed for high-throughput acquisition to enable precise and efficient analysis of material properties at the nanoscale.

Related work

Modern image analysis often involves techniques from ML and computer vision. When training an ML model, researchers sometimes perform *feature engineering* steps to provide the model with additional information. These steps may include transformations, such as binning numerical data³⁵, or augmentations, such as adding information about visual features in images³⁶. Deep learning is a subset of machine learning that relies on many-layer ("deep") neural networks and large amounts of training data. Deep learning can be advantageous because it can often learn patterns using simpler representations than those required for machine learning³⁷. While DL has demonstrated considerable success in a variety of fields, including materials science⁷, computer vision³⁸, medicine³⁹, physics⁴⁰, and others, the requirement for large amounts of training data can present a challenge.

Image segmentation involves partitioning images into regions of interest. Various ML and DL approaches have proved successful at this task, from convolutional neural networks (CNNs)⁴¹ to specialized architectures like U-net⁴²; a survey of DL techniques for image segmentation appears in⁴³. Specialized training and testing datasets to support image segmentation have been developed, such as Cityscapes⁴⁴, SYNTHIA⁴⁵, COCO⁴⁶, ADE20K⁴⁷, and the LIVECell dataset⁴⁸.

In microscopy and materials science, ML and DL methods have emerged as crucial tools for materials characterization^{3,4}. Segmentation models have been used to detect defects in steel using STEM images⁴⁹ and characterize X-ray computed tomography data of sintered alloys⁵⁰. Notable advances include the development of multiple output convolutional networks for multi-particle boundary detection⁵¹, comprehensive architecture comparisons for TEM image segmentation⁵², and genetic algorithm-based approaches for rapid morphological analysis⁵³. These developments demonstrate the field's progression toward automated analysis, though most focus on post-acquisition processing rather than acquisition optimization, which is the focus of this current work.

Our initial approach considered traditional ML or DL image segmentation techniques similar to those in^{5,6}. However, several key factors led us to explore more efficient alternatives. First, our grayscale nanoparticle images differ substantially from common ML datasets, which typically consist of color images with complex scenes. This distinction made direct adaptation of existing ML models impractical. While supervised transfer learning^{54,55} offered a potential solution for adapting existing models to our domain, the absence of ground-truth segmentation labels in our dataset and the prohibitive cost of creating them made this approach infeasible.

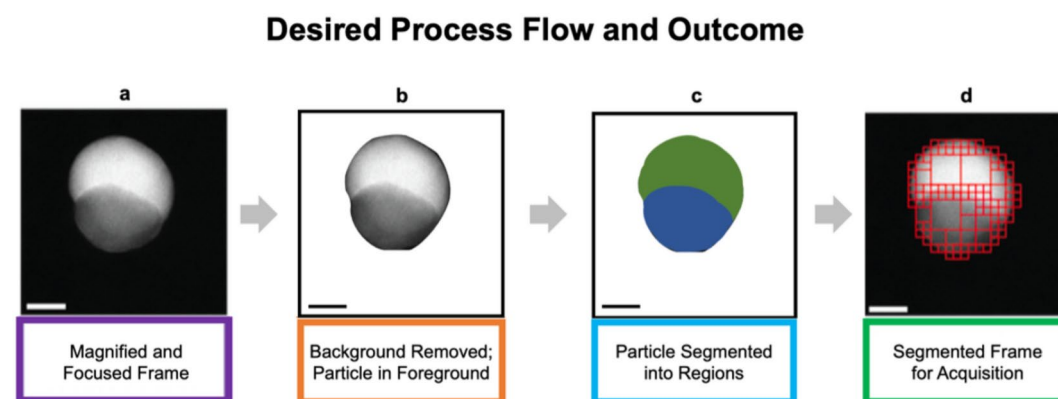


Fig. 1. (a) The original parent image; (b) a manually created image representing the desired outcome after the background removal step; (c) a manually created image representing the ideal segmentation regions for this image; (d) hand-drawn acquisition boxes overlaid on the parent image reflecting the desired coordinates for child images. Acquisition box size corresponds to the locations of features of interest: areas of varying intensity are captured with smaller boxes. Scale bars: 20 nm.

These constraints led us to explore unsupervised learning methods, specifically 1D k -means clustering⁵⁶. This approach offers several advantages: it groups similar data points without requiring labels, has well-established implementations in multiple libraries^{57–59}, and maintains computational efficiency. This choice aligns with recent trends in automated microscopy analysis^{7,8}, where unsupervised methods have shown promise in handling specialized scientific data.

For dividing segmented regions into acquisition boxes, we initially investigated geometric meshing techniques as in^{60,61}. However, two significant limitations emerged: first, standard meshing algorithms typically generate triangular grids⁶², incompatible with our requirement for square acquisition boxes; second, advanced adaptive refinement features often come embedded in comprehensive software suites such as⁶³, introducing unnecessary dependencies and potential performance overhead. These considerations led us to develop custom methods for generating acquisition coordinates (detailed in “Boxing algorithms”).

Our pipeline bridges the gap between initial microscopy imaging and high-resolution analysis techniques. The system generates adaptively-sized acquisition coordinates based on pixel intensity, optimizing data collection for regions of interest while minimizing unnecessary sampling. Our approach complements the development of other automated approaches in this field, such as⁹, and is expected to accelerate the characterization process. The implementation maintains high performance with minimal dependencies, ensuring versatility and longevity.

Datasets

Throughout this project, we relied on a unique set of high-angle annular dark-field (HAADF) images for development and evaluation. The images were acquired specifically to support the development of automated materials characterization. Data were collected on three dates using a JEOL JEM-ARM200CF scanning transmission electron microscope to mimic diverse laboratory conditions. The images were curated to represent a range of magnification levels and materials compositions, from one to five elements.

Since the first step in our overall nanoparticle characterization workflow involves a quality assessment, each image was analyzed by a human expert and labeled either “good” or “bad.” These labels correspond to whether or not the expert would choose to further analyze the particle under normal experimental conditions. These labels allowed us to train ML models in previous work to automate the initial quality assessment, as described in^{29–31}. Additional metadata in the file names provides unique identifiers for each image. More details about the datasets and collection settings are available in²⁹.

This work emulates the characterization step immediately following the quality check, where a particle has passed the assessment and been selected for in-depth analysis. As a result, we chose to filter our comprehensive dataset down to only the images that would reach this stage in the characterization workflow. Using the unique labels provided for each image, we focused on images labeled “good” with magnifications at or above 1,500,000 \times , resulting in a final dataset of 964 images for this work. The images were distributed across the three source datasets: 204 from the first, 399 from the second, and 361 from the third (Supplementary Table SI 1).

Methods

Our pipeline consists of three major components: image preprocessing, clustering, and acquisition box generation. Each component was developed with careful consideration of computational efficiency and effectiveness across diverse sample conditions. The following sections detail our procedures and design decisions for each step.

Image preprocessing steps

The first step in our pipeline involves downsizing the parent image to 128 \times 128 pixels. This step provides a direct comparison with the existing 128 \times 128 grid technique and allows us to standardize inputs across datasets (parent images have original resolutions of either 512 \times 512 or 1024 \times 1024, depending on the collection date). The downsizing step is illustrated in Fig. 2b.

Next, the image is cropped to remove the background and center the nanoparticle in the foreground. This step creates a smaller image, significantly reducing processing time. To prepare the image for cropping, we apply image enhancement techniques to reduce noise and highlight features. First, a sharpening mask is applied to the image (Fig. 2c), followed by a *GaussianBlur* filter (Fig. 2d) and an adaptive Gaussian threshold. This last step produces a binary mask with foreground and background regions, allowing us to calculate a bounding box that fully encloses the foreground pixels (Fig. 2e). Finally, the original 128 \times 128 image is cropped to the region enclosed by the bounding box, yielding a focused region of interest (Fig. 2f).

We then perform an additional step to further separate the foreground from the remaining background. We plot pixel intensity along the upper-left-to-lower-right diagonal in the cropped image, resulting in a profile that varies along with features in the image (Fig. 3a). We then compute a cutoff using information from the profile slope and total intensity range (Fig. 3b). During the design process, we implemented the same technique with the lower-left-to-upper-right diagonal, resulting in a second cutoff value. However, the differences between the two cutoffs were trivial in most cases (Fig. 3c), so we chose to use only information from the upper-left-to-lower-right diagonal to streamline the process.

Once the cutoff is calculated, we set the pixel values in the cropped image below the cutoff to 0, corresponding to the background, and set the remaining pixels, the foreground, to 1. This process produces a binary mask in the shape of the outline of the particle (Fig. 4b).

Since the nanoparticles in our application should always be convex, we perform an additional check at this step. First, we calculate the convex hull of the binary mask (Fig. 4c). This step returns a new binary mask that corrects any undesirable concavity in the original mask. We then use this mask to remove the background in the cropped image. The pixel locations in the cropped image corresponding to the mask background are set to 0, and the foreground pixels are brightened slightly to further differentiate them from the background. The amount by which the foreground is brightened is relative to the intensity range of the diagonal, as shown in Eq. (1):

Initial Preprocessing and Background Removal

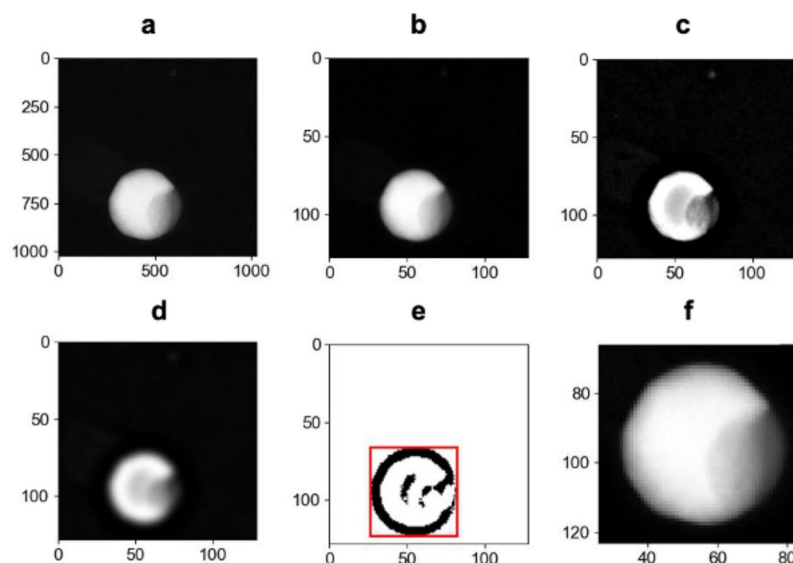


Fig. 2. The first six steps in the image preprocessing pipeline. Clockwise from top left: (a) the original image; (b) the original image resized to 128×128 pixels; (c) a sharpening mask applied to (b); (d) a Gaussian Blur applied to (c); (e) the binary mask resulting from Gaussian thresholding on (d), overlaid by the bounding box in red; (f) image (b) cropped to the coordinates of the bounding box in (e). The axis labels for all images show pixel dimensions; the axis bounds for (f) are relative to (b). The composition of this nanoparticle is $\text{Au}_{40}\text{Ag}_{10}\text{Cu}_{10}\text{Ni}_{40}$. The original image, shown in (a) has a magnification of $1,500,000\times$ and a resolution of 1024×1024 pixels.

$$\text{Foreground Offset} = \text{Ceiling}(\text{Range}(\text{Diagonal Pixel Intensities}) * 0.2) \quad (1)$$

Following these steps, the image is sharpened and blurred using the same techniques as before. The output image from this step becomes the input to the k -means clustering, as described next.

Clustering

Since we use 1D k -means clustering for image segmentation, we need to estimate the number of clusters in the image as precisely as possible. We experimented with several techniques and converged on a strategy that combines information about the number of elements in the particle composition and the peak pixel intensities in the image. This approach allows us to make use of chemical information specific to each particle. Looking ahead, this approach could be extended to novel materials compositions.

We used a histogram and scipy's *find_peaks*⁶⁴ tool to estimate the number of pixel intensity peaks. Since values of 0 correspond to the background and are not meaningful for this step, they were removed from the histogram. A *bin size* of 20 was used, and criteria was set to capture any peak with a prominence greater than 5% of the maximum value on the y-axis (Fig. 5).

Higher k values result in longer segmentation times, so it is crucial to identify the lowest possible k values that would produce reasonable segmentations. To find a good value of k for each image, we combine information about the number of peaks (p) calculated using the histogram and the number of elements (n) in the composition. Our procedure uses the following logic:

Procedure 1

If $(p < 2)$ or $(\text{abs}(p-n) \geq 2)$:

$k = n + 1$

Else:

$k = p + 1$

We use the resulting k value to segment our image using 1D k -means clustering (Fig. 6).

Boxing algorithms

We implemented two different algorithms for creating acquisition boxes. One approach involved finding and prioritizing the largest possible non-overlapping boxes within a given segmented region. The second approach was more efficient but did not prioritize the largest squares. The first strategy is described in Algorithm 1, and the second in Algorithm 2:

Pixel Intensity Based Background Removal

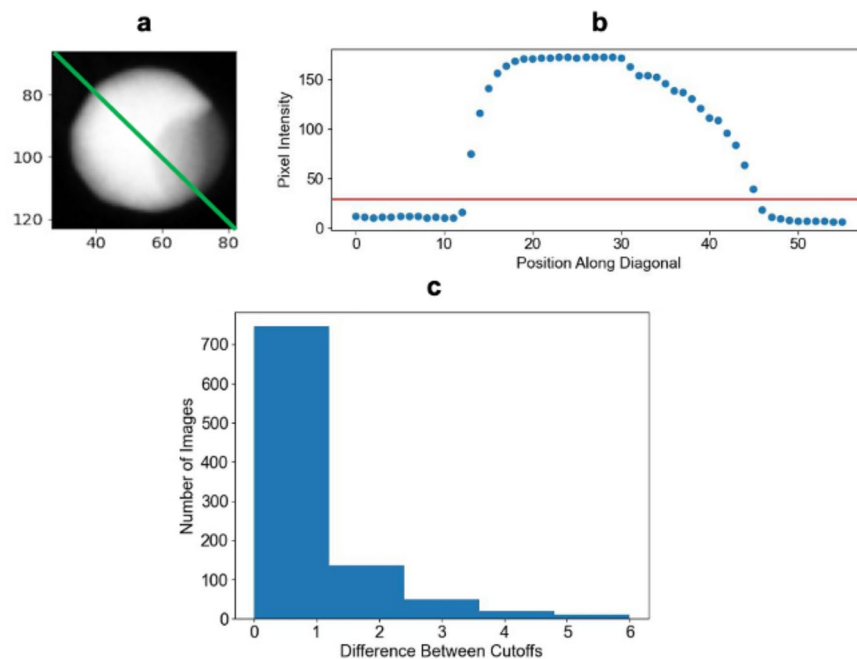


Fig. 3. To further separate the background from the foreground, we define a cutoff value by examining the pixel intensity of the upper-left-to-lower-right diagonal of the cropped image. The cropped image is shown in (a) with the diagonal overlaid in green for reference; (b) a plot of the pixel intensities along this diagonal, with the cutoff value as a red horizontal line; (c) a histogram of the difference in intensity values between the upper-left-to-lower-right and upper-right-to-lower-left diagonals across the entire dataset.

Further Preprocessing

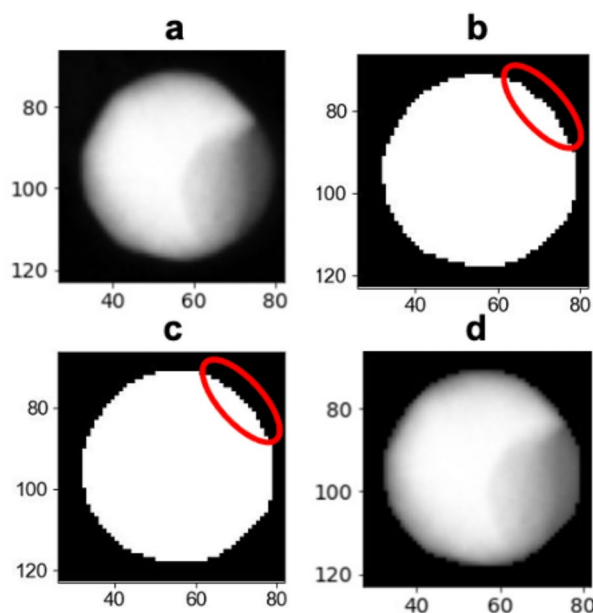


Fig. 4. (a) cropped image; (b) binary mask of (a) after the background removal procedure in Fig. 3; (c) convex hull of (b); (d) the result of filtering (a) based on the mask in (c) and brightening the foreground by the amount in Eq. (1). Red ovals highlight the difference between the binary mask and the convex hull.

Pixel Intensity Based k Value Prediction

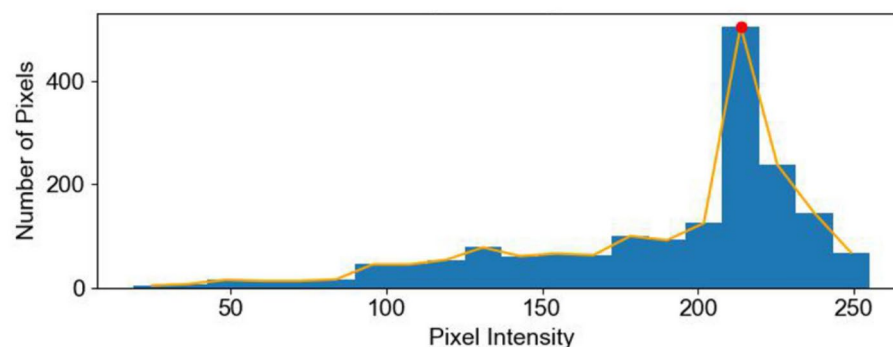


Fig. 5. The process identifies one pixel intensity peak for the particle shown in Figs. 2, 3 and 4.

Segmentation Results

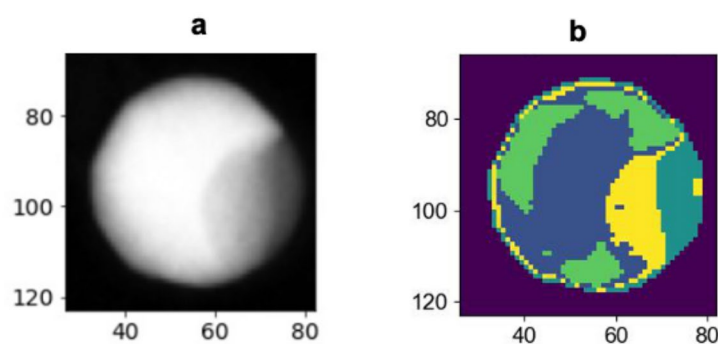


Fig. 6. (a) The cropped image, shown for reference; (b) final segmentation result after applying k-means clustering with $k=5$.

Algorithm 1

Description: Returns a list of squares T that completely fill some segmented region R inside an image array A , except the background region. Sorts the squares to prioritize the largest non-overlapping boxes. Each element of T contains upper-left-hand coordinates x, y and size s .

Input: A , a mask array the same size as the image; for some given element $a \in A$, $a=1$ if the pixel at that location $\in R$; 0 otherwise. Let the element at position x, y be denoted $A[x, y]$. Let the lower-right-hand corner of A be denoted as $A[\max_x, \max_y]$. Let Q be a queue containing the x and y coordinates of all initial nonzero values of A , where the order of the coordinates is the same as when iterating across columns and then down rows of A . The maximum allowed box size is specified as m .


```

 $F \leftarrow \emptyset$ 
 $T \leftarrow \emptyset$ 

1. while  $Q$  is not empty do
2.    $q \leftarrow Q.get()$ 
3.   if  $A[q_x, q_y] \neq 0$  then
4.      $V \leftarrow$  all nonzero elements of  $A$ 
5.     for all  $[v_x, v_y] \in V$  do
6.        $max\_box\_len \leftarrow \min(m, max_x - v_x, max_y - v_y)$ 
7.        $P \leftarrow A[v_x:v_x+max\_box\_len, v_y:v_y+max\_box\_len]$ 
8.       if all  $[p_x, p_y] \in P == 1$  then
9.          $s \leftarrow max\_box\_len$ 
10.      else
11.        for  $i \in [1, max\_box\_len-1]$  do
12.           $U \leftarrow P[:, i:i]$ 
13.          if not all  $[u_x, u_y] \in U == 1$  then
14.             $s \leftarrow i-1$ 
15.          end if
16.        end for
17.         $s \leftarrow i$ 
18.      end if
19.       $F \leftarrow F.append(v_x, v_y, s)$ 
20.    end for
21.    sort  $F$  by  $s$  largest to smallest
22.     $T \leftarrow T.append(F_0)$ 
23.     $A[F_{0x}:F_{0x}+F_{0s}, F_{0y}:F_{0y}+F_{0s}] \leftarrow 0$ 
24.    for all  $f \in F$  do
25.      if box  $f$  of  $F$  does not overlap
        some box already in  $T$  then
26.         $A[F_{fx}:F_{fx}+F_{fs}, F_{fy}:F_{fy}+F_{fs}] \leftarrow 0$ 
27.         $T \leftarrow T.append(F_f)$ 
28.      end if
29.    end for
30.  end if
31. end while
32. return  $T$ 

```

initialize

consider all nonzero values in A
 # return next value of Q , denote as q
 # check element of interest is nonzero
 # calculate nonzero elements of A
 # find largest box at each location
 # find maximum possible length
 # limit A to maximum possible size
 # if all pixels of P are in region R
 # use largest possible size

loop from 1 to max_box_len-1
 # increase box size each time
 # if any pixels outside region
 # use previous i

if loop terminates, use max_box_len-1
 # append box values to F

consider largest boxes first
 # add largest box in F to T
 # set area enclosed by largest box to 0
 # add any non-overlapping boxes in F to T

set area enclosed by box to 0
 # append values to T

Algorithm 2

Description: Returns a list of squares F that completely fill some segmented region R inside an image array A , except the background region. Each element of F contains upper-left-hand coordinates x, y and size s .

Input: A , a mask array the same size as the image; for some given element $a \in A$, $a = 1$ if the pixel at that location $\in R$; 0 otherwise. Let the element at position x, y be denoted $A[x, y]$. Let the lower-right-hand corner of A be denoted as $A[max_x, max_y]$. Let Q be a queue containing the x and y coordinates of all initial nonzero values of A , where the order of the coordinates is the same as when iterating across columns and then down rows of A . The maximum allowed box size is specified as m .

```

 $F \leftarrow \emptyset$                                      # initialize

1. while  $Q$  is not empty do                       # consider all nonzero values in region
2.    $q \leftarrow Q.get()$                          # return next value of  $Q$ , denote as  $q$ 
3.   if  $A[q_x, q_y] \neq 0$  then                   # check element of interest is nonzero
4.      $max\_box\_len \leftarrow \min(m, max_x - q_x, max_y - q_y)$  # find maximum possible length
5.      $P \leftarrow A[q_x:q_x+max\_box\_len, q_y:q_y+max\_box\_len]$  # limit  $A$  to maximum possible size
6.     if all  $[p_x, p_y] \in P == 1$  then           # if all pixels of  $P$  are in region  $R$ 
7.        $s \leftarrow max\_box\_len$                  # use largest possible size
8.     else
9.       for  $i \in [1, max\_box\_len-1]$  do           # loop from 1 to  $max\_box\_len-1$ 
10.         $U \leftarrow P[i:, :i]$                  # increase box size each time
11.        if not all  $[u_x, u_y] \in U == 1$  then   # if any pixels outside region
12.           $s \leftarrow i-1$                      # use previous  $i$ 
13.        end if
14.      end for
15.       $s \leftarrow i$                            # if loop terminates, use  $max\_box\_len-1$ 
16.    end if
17.     $F \leftarrow F.append(p_x, p_y, s)$           # append values to  $F$ 
18.     $A[p_x:p_x+s, p_y:p_y+s] \leftarrow 0$        # set values of enclosed area to 0
19.  end if
20. end while
21. return  $F$ 

```

In consultation with domain experts on our team, we decided to cap the size of the maximum possible box to 20% of the shorter side of the cropped image. For example, a cropped image of size 100×80 pixels would contain acquisition boxes up to a maximum size of $80 \times 0.2 = 16$ pixels. Setting this upper bound allowed us to standardize our results and prevent detail loss in large images. It also made Algorithm 2 the clear choice for this application, although we note that Algorithm 1 may be the preferred choice in other situations. Algorithm 2 was used for the following results. An example application of Algorithm 2 is shown in Fig. 7; additional examples of the process are in Supplementary Figs. SI 1–6.

Results

After the final design decisions were made, we ran the code on all 964 images in our dataset. We gathered timing information from these experiments and performed a visual inspection of the results.

Timing and performance

To standardize the segmentation results on successive code runs, we set a random seed and verified that the segmentation and boxing results were unchanged between runs when the seed was applied. We used python's

Boxes Calculated by Method

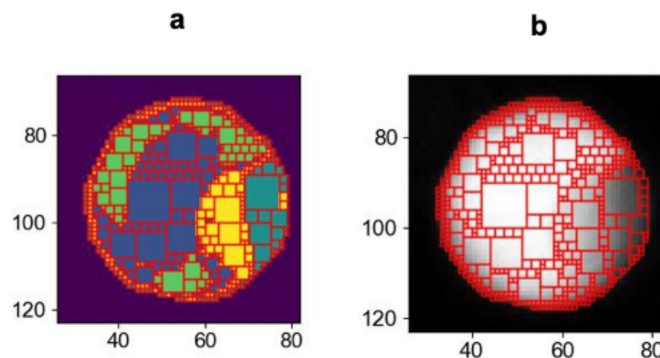


Fig. 7. (a) Acquisition boxes calculated by Algorithm 2 overlaid on the segmentation result in Fig. 6. (b) The same boxes overlaid on the cropped image.

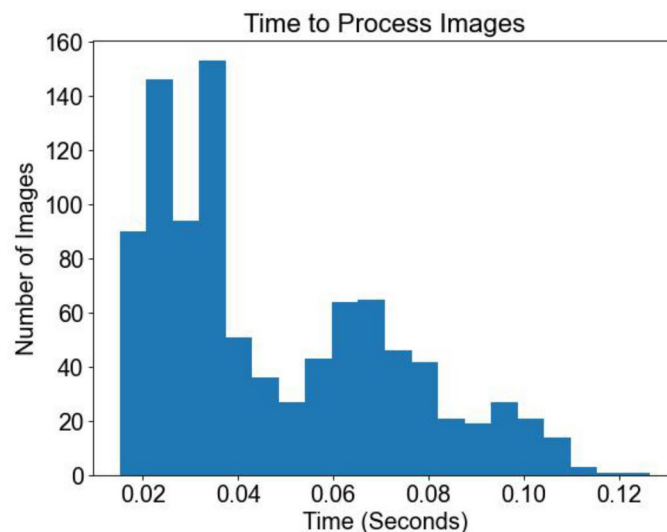


Fig. 8. Histogram of the median processing times for our dataset, when each image is run 30 times. In other words, we processed each image 30 times, then calculated the median processing time for that image. This figure shows the distribution of median times over the entire dataset. This processing time includes the preprocessing, segmentation, and box calculation steps.

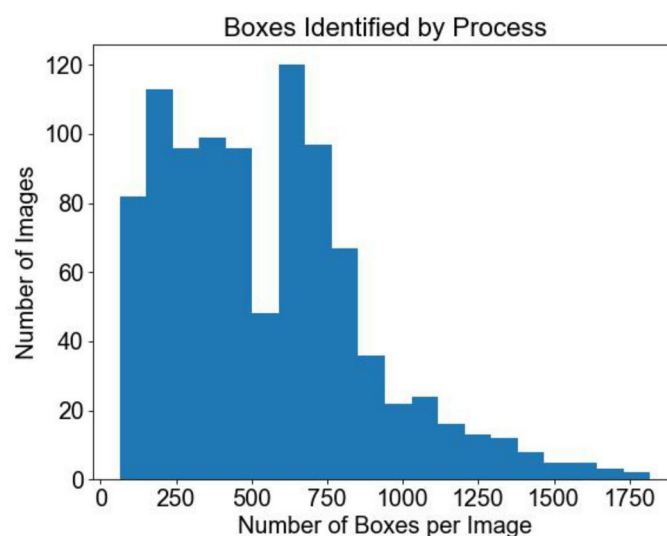


Fig. 9. The distribution of boxes identified by the procedure across the dataset.

*cProfiler*⁶⁵ to identify performance bottlenecks and achieved significant speedups between successive versions of the code. The timing tests described below were conducted on a 2020 MacBook Pro, running macOS Sonoma 14.5. Run times in the laboratory may vary, depending on the software environment and other processes on the machine.

To account for timing variability, we ran the code on each image 30 times while calculating the elapsed processing time. We then calculated the median of those 30 values for each image. In the following calculations, we examined the distribution of these median values across the entire dataset. The average of the median image processing times across the entire dataset was 0.05 s, with a maximum of 0.13 s and a minimum of 0.02 s. The median value was 0.04 s. The distribution of these median processing times is shown in Fig. 8:

Because the random seed guarantees that segmentation and boxing results are the same every time, we report the number of boxes calculated on a single run of the code. The number of boxes varied between 63 and 1819, with a mean value of 551.2 and a median of 493.0. Since the original procedure captured 16,384 child images for each parent image, our process results in 9.0 times fewer boxes in the worst-case scenario, 260.1 times fewer in the best case, and 29.7 times fewer in the average case. This streamlined process saves time and data storage space. The distribution of the number of calculated boxes across the entire dataset is shown in Fig. 9:

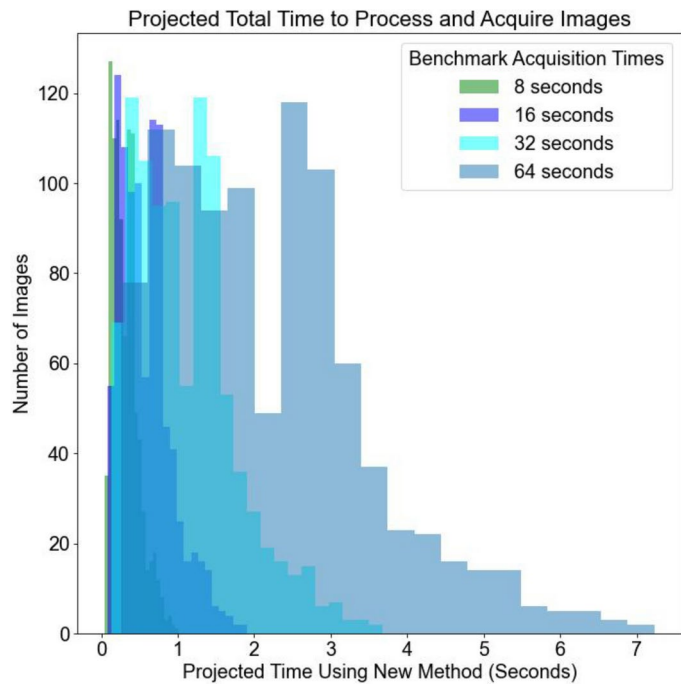


Fig. 10. Projected total time to process and acquire images, using Eq. (2), for benchmark acquisition methods ranging from 8 to 64 s. To calculate these times, the median runtime of 30 trials was used for each image.

Current Baseline	Projected Total Time <i>T</i> Using Proposed Method (median value over 30 trials; times in seconds)				
	Mean	Minimum	Maximum	Median	Speedup of Mean Relative to Baseline
8 s	0.32	0.05	1.01	0.29	25.0
64 s	2.20	0.26	7.23	1.97	29.1

Table 1. Timing and performance statistics across the entire dataset.

To benchmark our process, we calculate the projected total time needed to process an image and capture 4D-STEM data using known laboratory acquisition times. For an image, we can represent the total time to process and acquire data using Eq. (2), where T is the total time, t_{seg} is the time for the segmentation and boxing process, n_b is the number of boxes, and t_{acq} is the projected acquisition time per data point.

$$T = t_{seg} + n_b * t_{acq}.$$

Our benchmark 4D-STEM acquisition methods take 8, 16, 32, and 64 s for a single image and capture 16,834 individual data points. We calculate that it takes $t_{acq} = \frac{s_b}{16,834}$ to acquire a single data point, where s_b is a known benchmark time. For example, given a current acquisition process that takes 8 s per image, we calculate $t_{acq} = \frac{8seconds}{16,834points} = 0.488$ ms per data point. To show how the speedup changes for each of these values, we calculate the distribution of T expected for each of these values (Fig. 10).

For these calculations, we again used the median timing value over 30 trials for each image and then calculated statistics across the entire dataset (Table 1).

We note that the acquisition time portion ($n_b * t_{acq}$) dominates the total projected runtimes, and the proposed preprocessing, segmentation, and boxing time (t_{seg}) is a relatively insignificant component of the total. Note that t_{seg} is independent of the baseline acquisition method, i.e., it is the same for both the 8 s and 64 s baselines.

Visual inspection

Running the code on our dataset generates 964 distinct pipeline results, which we used to evaluate overall performance. We used the following criteria to judge each image: (1) the bounding box should fully enclose the entire particle, including any faint edges; (2) the segmentation step should separate any areas in the image that differ in pixel intensity; (3) the returned acquisition boxes should reflect regions of interest within the original image. Using these metrics, we performed a visual inspection on a single run of the code. We identified 925 of the final images as meeting all criteria, providing an overall success rate of 96.0%. In the absence of ground-truth labels, we acknowledge that this technique is subjective and depends on individual judgment. However, example

results with ambiguous outputs were discussed with domain experts in our team before this final figure was calculated.

Conclusion

We present an efficient, automated image segmentation pipeline that addresses a critical bottleneck in high-throughput nanoparticle characterization. The pipeline is expected to process images 25.0 to 29.1 times faster than baseline methods and operates without human input. We evaluated the results of our pipeline on over 900 diverse images and report a 96.0% success rate based on expert-validated criteria. The system effectively handles variations in focus, magnification, and contrast, demonstrating robust performance across different experimental conditions. Moreover, our approach is expected to reduce the number of acquisition points by an average factor of 29.7, significantly saving storage space. The pipeline's distinctive features include adaptive sizing of acquisition boxes and integration of particle composition information in the segmentation process. Our code was optimized for speed and designed to contain minimal dependencies, ensuring long-term efficiency and stability. Future work could involve extending this methodology to additional microscopy datasets and performing detailed studies of real-time laboratory deployment. This automated pipeline represents a significant step toward fully automated materials characterization, reducing human intervention in routine preprocessing tasks and accelerating data acquisition.

Data availability

The authors confirm that data supporting the findings of this study are available within the article and its supplementary materials.

Code availability

The proposed algorithm is detailed in the article. Further, the implementation (code) of the proposed algorithm will be available at https://github.com/alexandrday-nu/particle_segmentation from the date of publication, subject to a copyright license that allows for non-commercial, educational, and research use.

Received: 21 March 2024; Accepted: 25 April 2025

Published online: 17 May 2025

References

- Xu, W. & LeBeau, J. M. A deep convolutional neural network to analyze position averaged convergent beam electron diffraction patterns. *Ultramicroscopy* **188**, 59–69 (2018).
- Zhang, C., Feng, J., DaCosta, L. R. & Voyles, P. M. Atomic resolution convergent beam electron diffraction analysis using convolutional neural networks. *Ultramicroscopy* **210**, 112921 (2020).
- Zhang, C., Han, R., Zhang, A. R. & Voyles, P. M. Denoising atomic resolution 4D scanning transmission electron microscopy data with tensor singular value decomposition. *Ultramicroscopy* **219**, 113123 (2020).
- Munshi, J. et al. Disentangling multiple scattering with deep learning: application to strain mapping from electron diffraction patterns. *Npj Computat. Mater.* **8**, 254 (2022).
- Oxley, M. P. et al. Probing atomic-scale symmetry breaking by rotationally invariant machine learning of multidimensional electron scattering. *Npj Computat. Mater.* **7**, 65 (2021).
- Bruefach, A., Ophus, C. & Scott, M. C. Robust design of semi-automated clustering models for 4D-STEM datasets. *APL Mach. Learn.* **1**, 016106 (2023).
- Yuan, R., Zhang, J., He, L. & Zuo, J.-M. Training artificial neural networks for precision orientation and strain mapping using 4D electron diffraction datasets. *Ultramicroscopy* **231**, 113256 (2021).
- Shi, C. et al. Uncovering material deformations via machine learning combined with four-dimensional scanning transmission electron microscopy. *Npj Computat. Mater.* **8**, 1–9 (2022).
- Bridger, A., David, W. I. F., Wood, T. J., Danaie, M. & Butler, K. T. Versatile domain mapping of scanning electron nanobeam diffraction datasets utilising variational autoencoders. *Npj Computat. Mater.* **9**, 14–18 (2023).
- Kluender, E. J. et al. Catalyst discovery through megalibraries of nanomaterials. *Proc. Natl. Acad. Sci.* **116**, 40–45 (2019).
- Agrawal, A. & Choudhary, A. Perspective: materials informatics and big data: realization of the “fourth paradigm” of science in materials science. *APL Mater.* <https://doi.org/10.1063/1.4946894> (2016).
- Dimiduk, D. M., Holm, E. A. & Niezgoda, S. R. Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering. *Integr. Mater. Manuf. Innov.* **7**, 157–172 (2018).
- Jha, D. et al. Extracting grain orientations from EBSD patterns of polycrystalline materials using convolutional neural networks. *Microsc. Microanal.* **24**, 497–502 (2018).
- Gupta, V. et al. Cross-property deep transfer learning framework for enhanced predictive analytics on small materials data. *Nat. Commun.* <https://doi.org/10.1038/s41467-021-26921-5> (2021).
- Jha, D. et al. ElemNet: Deep learning the chemistry of materials from only elemental composition. *Sci. Rep.* <https://doi.org/10.1038/s41598-018-35934-y> (2018).
- Choudhary, K. et al. Recent advances and applications of deep learning methods in materials science. *NPI Comput. Mater.* **8**, 1–26 (2022).
- Agrawal, A. & Choudhary, A. Deep materials informatics: Applications of deep learning in materials science. *MRS Commun.* **9**, 779–792 (2019).
- Jha, D. et al. Enabling deeper learning on big data for materials informatics applications. *Sci. Rep.* <https://doi.org/10.1038/s41598-021-83193-1> (2021).
- Kalinin, S. et al. Machine learning for automated experimentation in scanning transmission electron microscopy. *Npj Computat. Mater.* <https://doi.org/10.1038/s41524-023-01142-0> (2023).
- DeCost, B. et al. Scientific AI in materials science: A path to a sustainable and scalable paradigm. *Machine Learning: Science and Technology* **1**, 033001 (2020).
- Mehta, A. et al. Materials science in the artificial intelligence age: high-throughput library generation, machine learning, and a pathway from correlations to the underpinning physics. *MRS Commun.* **9**, 821–838 (2019).
- Choudhary, K. et al. The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. *Npj Computat. Mater.* **6**, 2057–3960 (2022).

23. Sbailò, L., Fekete, A., Ghiringhelli, L. & Scheffler, M. The NOMAD artificial-intelligence toolkit: Turning materials-science data into knowledge and understanding. *npj Computat. Mater.* <https://doi.org/10.1038/s41524-022-00935-z> (2022).
24. Ziatdinov, M., Ghosh, A., Wong, C. Y. & Kalinin, S. V. AtomAI framework for deep learning analysis of image and spectroscopy data in electron and scanning probe microscopy. *Nat. Mach. Intell.* **4**, 1101–1112 (2022).
25. Liu, R. et al. A predictive machine learning approach for microstructure optimization and materials design. *Sci. Rep.* <https://doi.org/10.1038/srep11551> (2015).
26. Cang, R., Li, H., Yao, H., Jiao, Y. & Ren, Y. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. *Comput. Mater. Sci.* **150**, 212–221 (2018).
27. Yang, Z. et al. Deep learning approaches for mining structure-property linkages in high contrast composites from simulating datasets. *Comput. Mater. Sci.* **151**, 278–287 (2018).
28. Pyzer-Knapp, E. O. et al. Accelerating materials discovery using artificial intelligence, high performance computing and robotics. *npj Computational Materials* **8**, (2022).
29. Day, A. L. et al. Machine learning-enabled image classification for automated electron microscopy. *Microsc. Microanal.* **30**, 456–465 (2024).
30. Wahl, C. B. et al. Machine learning enabled image classification for automated data acquisition in the electron microscope. *Microscopy Microanal.* **29**, 1909–1910 (2023).
31. Day, A. L. et al. Automated nanoparticle image processing pipeline for AI-driven materials characterization. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 4462–4469. (2024).
32. Ophus, C. Four-dimensional scanning transmission electron microscopy (4D-STEM): From scanning nanodiffraction to ptychography and beyond. *Microsc. Microanal.* **25**, 563–582 (2019).
33. Girão, A. V., Caputo, G., and Ferro, M. C. Chapter 6 - Application of Scanning Electron Microscopy–Energy Dispersive X-Ray Spectroscopy (SEM-EDS). In *Comprehensive Analytical Chemistry*. Rocha-Santos, T. A. P., & Duarte, A. C. (Eds.), **75**, 153–168. (2017).
34. Hofer, F., Schmidt, F. P., Grogger, W. & Kothleitner, G. Fundamentals of electron energy-loss spectroscopy. *IOP Conf. Ser. Mater. Sci. Eng.* **109**, 1012007 (2016).
35. Feature Engineering for Machine Learning. [Online]. <https://learning.oreilly.com/library/view/feature-engineering-for/9781491953235/>. Accessed November 22, 2024.
36. Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* **60**, 91–110 (2004).
37. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
38. Chai, J., Zeng, H., Li, A. & Ngai, E. W. T. Deep learning in computer vision: a critical review of emerging techniques and application scenarios. *Mach. Learn. Appl.* **6**, 100134 (2021).
39. Esteva, A. et al. A guide to deep learning in healthcare. *Nat. Med.* **25**, 24–29 (2019).
40. Glaser, C., McAleer, S., Stjärnholm, S., Baldi, P. & Barwick, S. W. Deep-learning-based reconstruction of the neutrino direction and energy for in-ice radio detectors. *Astropart. Phys.* **145**, 102781 (2023).
41. Shelhamer, E., Long, J. & Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 640–651 (2017).
42. Ronneberger, O., Fischer, P., & Brox, T. U-Net: convolutional networks for biomedical image segmentation. *The 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*. **9351**, 234–241 (2015).
43. Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. Image segmentation using deep learning: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **1** (2021).
44. M. Cordts et al., The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3213–3223 (2016).
45. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., & Lopez, A. M. The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3234–3243 (2016).
46. Lin, T.-Y. et al., Microsoft COCO: common objects in context. *Computer Vision – ECCV 2014*. 740–55 (2014).
47. Zhou, B. et al. Semantic understanding of scenes through the ADE20K dataset. *Int. J. Comput. Vision* **127**, 302–321 (2019).
48. Christoffer, E. et al. LIVECell—a large-scale dataset for label-free live cell segmentation. *Nat. Methods* **18**, 1038–1045 (2021).
49. Roberts, G. et al. Deep learning for semantic segmentation of defects in advanced STEM images of steels. *Sci. Rep.* **9**, 12744 (2019).
50. Varoto, L. et al. 3D microstructure characterization of Cu25Cr solid state sintered alloy using X-ray computed tomography and machine learning assisted segmentation. *Mater. Charact.* **203**, 113107 (2023).
51. Oktay, A. B. & Gurses, A. Automatic detection, localization and segmentation of nano-particles with deep learning in microscopy images. *Micron* **120**, 113–119 (2019).
52. Saaim, K. M., Afridi, S. K., Nisar, M. & Islam, S. In search of best automated model: Explaining nanoparticle TEM image segmentation. *Ultramicroscopy* **233**, 113437 (2022).
53. Lee, B. et al. Statistical characterization of the morphologies of nanoparticles through machine learning based electron microscopy image analysis. *ACS Nano* **14**, 17125–17133 (2020).
54. Taylor, M. E. & Stone, P. Transfer learning for reinforcement learning domains: A Survey. *J. Mach. Learn. Res.* **10**, 1633–1685 (2009).
55. Pan, S. J. & Yang, Q. A Survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **22**, 1345–1359 (2010).
56. Macqueen, J. Some methods for classification and analysis of multivariate observations. *Proc. Fifth Berkeley Sympos. Math. Stat. Prob.* **1**, 281–297 (1967).
57. k-means clustering. *MATLAB Documentation* [Online]. <https://www.mathworks.com/help/stats/kmeans.html>. Accessed November 22, 2024.
58. OpenCV: K-Means Clustering in OpenCV. *Open Source Computer Vision Library*. [Online]. https://docs.opencv.org/3.4/d1/d5c/tutorials_py_kmeans_opencv.html. Accessed November 22, 2024.
59. sklearn.cluster.KMeans. *scikit-learn 1.4.0 Documentation*. [Online]. <https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.KMeans.html>. Accessed November 22, 2024.
60. Blacker, T. D. & Stephenson, M. B. Paving: A new approach to automated quadrilateral mesh generation. *Int. J. Numer. Meth. Eng.* **32**, 811–847 (1991).
61. Berger, M. & Oliger, J. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.* **53**, 484–512 (1984).
62. Bern M. & Eppstein, D. Mesh generation and optimal triangulation. *Computing in Euclidean Geometry*, 47–123. (1995).
63. Permann, C. J. et al. MOOSE: Enabling massively parallel multiphysics simulation. *SoftwareX* **11**, 100430 (2020).
64. scipy.signal.find_peaks. *SciPy v1.11.4 Manual*. [Online]. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html. Accessed November 22, 2024.
65. The Python Profilers. *Python 3.12.1 Documentation*. [Online]. <https://docs.python.org/3/library/profile.html#module-cProfile>. Accessed November 22, 2024.

Acknowledgements

This work is supported in part by: Northwestern Center for Nanocombinatorics; National Institute of Standards

and Technology (NIST) award 70NANB24H136; Department of Energy (DOE) award DE-SC0021399; NSF awards CMMI-2053929, OAC-2331329; the Sherman Fairchild Foundation, Inc.; the Toyota Research Institute, Inc.; and the Army Research Office grant W911NF-23-1-0141. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-2234667. This work made use of Northwestern University's NUANCE Center, which has received support from the SHyNE Resource (NSF ECCS-2025633), the IIN, and Northwestern's MRSEC program (NSF DMR-2308691). This research was supported in part through the computational resources and staff contributions provided for the Quest high performance computing facility at Northwestern University which is jointly supported by the Office of the Provost, the Office for Research, and Northwestern University Information Technology.

Author contributions

ALD led the workflow design, wrote the analysis code and conducted experiments and visual inspections. CW collected, labeled, and organized the data. AA provided expertise and guidance in machine learning and contributed to major pipeline design decisions. RdR and CW provided domain-specific expertise, reviewed sample outputs, and suggested key improvements. AC, AA, WKL, VPD, and CAM provided supervision and helped with manuscript preparation. ALD wrote the manuscript with AA, CW, RdR, MNTK and YL. All authors read and approved the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-025-01337-z>.

Correspondence and requests for materials should be addressed to A.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025