Developing a machine learning system for the unsupervised classification of defects in Scanning Tunnelling Microscopy images

By

Jay Patel

Supervisor: Dr. Steven Schofield

Department of Physics and Astronomy
UNIVERSITY COLLEGE LONDON

Date of submission: April 2, 2025
Word count: 7469

UCL DEPARTMENT OF PHYSICS & ASTROMOMY

Submission of coursework for Physics and Astronomy course PHAS0097/PHAS0096/PHAS0048

Please sign, date and return this form with your coursework by the specified deadline.

DECLARATION OF OWNERSHIP

I confirm that I have read and understood the guidelines on plagiarism, that I understand the meaning of plagiarism and that I may be penalised for submitting work that has been plagiarised.

I confirm that all work will also be submitted electronically and that this can be checked using the JISC detection service, Turnitin®.

I declare that all material presented in the accompanying work is entirely my own work except where explicitly and individually indicated and that all sources used in its preparation and all quotations are clearly cited.

Should this statement prove to be untrue, I recognise the right of the Board of Examiners to recommend what action should be taken in line with UCL's regulations.

Signed Patel

PrintName Jay Patel

Dated 2/4/2025

Title	Date Received	Examiner	Examiner's Signature	Mark

Abstract

This report details the development and evaluation of a two-stage image analysis pipeline designed for the characterisation of defects in Scanning Tunneling Microscopy (STM) images. The first stage involves general segmentation using a convolutional autoencoder (AE1) that compresses STM images into a 32-dimensional latent space, which is then clustered using K-means to identify structural similarities. The second stage focuses specifically on defect clustering, utilising a secondary autoencoder (AE2) that captures detailed defect features in a reduced 8-dimensional latent space, with clustering performed by Gaussian Mixture Models (GMMs). Results indicate that the 32-neuron model was optimal for balancing reconstruction accuracy and clustering efficiency, as evidenced by low Davies-Bouldin indices and coherent clustering in both standard and defect-specific latent spaces. However, the model exhibited limitations in handling atomic step edges, highlighting a potential area for future enhancement with edge-detection techniques.

Acknowledgements

I would like to express my sincere gratitude to the London Centre for Nanotechnology (LCN) for the provision of high-performance computing facilities, which played a crucial role in the data processing and simulation tasks described in this report. Special thanks are also extended to Dr. Steven Schofield for his invaluable guidance and expert insights throughout the duration of this project.

Contents

List of Figures

1. Introduction

Image recognition is a core task in machine learning (ML), with applications ranging from identifying types of pastries or cancer cells to algorithms for self-driving vehicles and facial recognition in security systems [1, 2, 3]. It has since become a standard benchmark task in ML research. Traditional image processing techniques, such as Gaussian filtering, edge detection, and thresholding, rely on predefined rules and manually extracted features. While effective for some applications, these methods struggle with modern scientific image data's complexity, noise, and variability. As a result, they are poorly suited to high-resolution, atomic-scale images such as those produced by scanning tunnelling microscopy (STM).

STM is a powerful technique that enables the generation of high-resolution images of surfaces at the atomic scale. Since its invention in 1982 by Binnig and Rohrer [4], STM has revolutionised the field of nanotechnology by providing unprecedented insights into the behaviour of individual atoms. Its applications span disciplines, including semiconductor fabrication, materials science, and surface physics [4, 5]. The ability to visualise atomic-scale features has proven crucial to the development of advanced technologies, from quantum electronics to novel materials.

However, STM image analysis remains labour-intensive, requiring expert interpretation to identify and classify atomic-scale features such as adsorbed atoms and molecules. The sheer volume of data produced makes manual analysis impractical, highlighting the need for automated and efficient analysis techniques. In this context, ML techniques present an exciting opportunity to automate the identification and classification of atomic features in STM images. By leveraging the power of deep learning, this project aims to accelerate STM image analysis and facilitate the extraction of meaningful insights from large datasets, enabling more efficient progress in nanotechnology research.

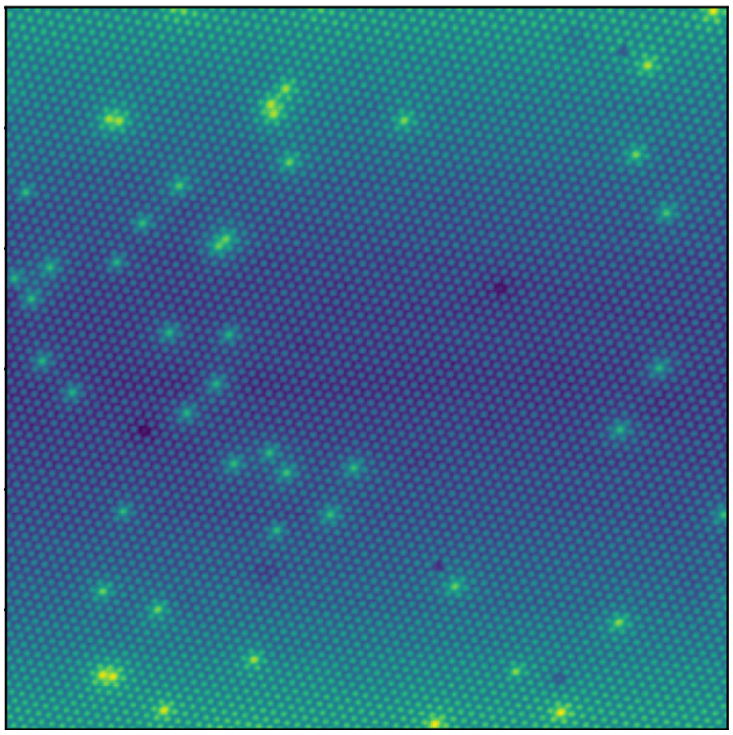


Figure 1.1: Example of a raw scanning tunnelling microscopy (STM) image used in this project. Bright spots indicate elevated topographic features, likely corresponding to adsorbed atoms or surface defects, while the regular grid pattern represents the underlying atomic lattice of the substrate.

1.1. Our data

STM produces complex, high-resolution images with atomic precision. In this project, we analyse STM images of surfaces relevant to the development of molecular precursors for single-atom quantum-electron devices. These images contain adsorbed atom groups whose arrangement is critical to the nanoscale properties of the material. Variability in atomic structure, image noise, and distortions pose significant challenges for automated analysis.

Because our dataset is unlabelled, we adopt a fully unsupervised learning framework. The primary goal is to identify and classify atomic-scale features, particularly adsorbed groups, using unsupervised ML techniques that can uncover structure without human-annotated labels. This is achieved by first using autoencoders (AEs) to reduce image dimensionality and extract meaningful features, followed by clustering algorithms such as K-means to group structurally similar regions. This enables automated categorisation of atomic features across STM images and supports scalable, high-throughput analysis.

1.2. Neural networks

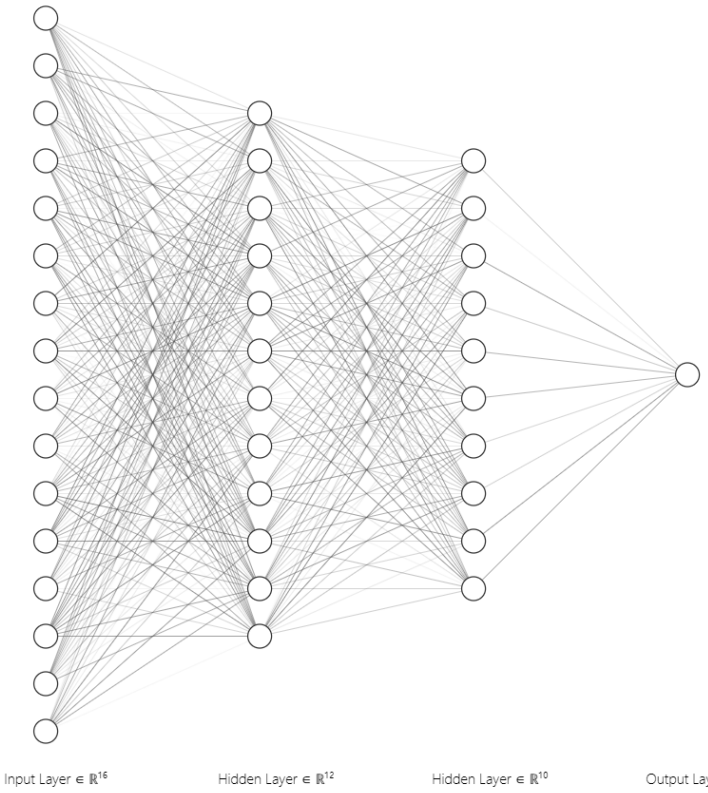


Figure 1.2: A generic example of a fully connected feed-forward neural network. Although A generic example of a fully connected feed-forward neural network. While not directly employed in this project, such architectures underpin more sophisticated models like convolutional neural networks and autoencoders. The opacity of the connections reflects the relative magnitude of the weights, with darker lines indicating stronger connections between neurons.

Neural networks are computational models inspired by the structure and function of biological neurons, designed to recognise patterns and make decisions. These networks consist of layers of interconnected neurons, each of which processes data through a series of weights and activation functions. In a typical feed-forward neural network (FNN), the structure includes an input layer, one or more hidden layers, and an output layer (fig. 1.2).

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1.1}$$

f represents the activation function applied to the summation of weighted inputs x_i and biases b. The activation function determines the neuron's output, which is passed to neurons in the next layer of the network. Neurons are adjusted during training to optimise the network's predictions. Training involves optimising these parameters to minimise error,

commonly using back-propagation.

A typical neural network consists of three main layers: the input, hidden, and output layers (fig. 1.2). The input layer receives the initial data, while the hidden layers progressively transform the data to capture more abstract features. Finally, the output layer produces the network's prediction or decision based on the transformed input.

Neural networks are trained by adjusting their parameters, such as weights and biases, using iterative learning processes. The most common method of training involves back-propagation, which adjusts weights in response to errors made by the network. This iterative adjustment allows the network to gradually improve its performance as it encounters more data, ultimately making more accurate predictions.

Various activation functions are used in neural networks depending on the task. Common examples include the Rectified Linear Unit (ReLU) and the sigmoid activation function (fig. 1.3). The ReLU function is often used in hidden layers to introduce non-linearity and allow the network to learn more complex patterns. The sigmoid function is typically used in the output layer for binary classification tasks, as it maps the output to a range between 0 and 1.

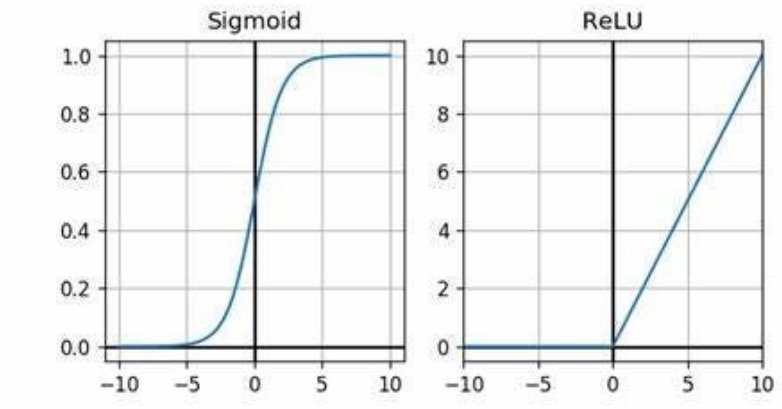


Figure 1.3: Comparison of activation functions. The sigmoid function (left) squashes input values into the range $[0, 1]$, while the ReLU function (right) outputs the input value directly for positive inputs and zero otherwise.

Feed-forward, fully connected neural networks consist of an input layer, one or more hidden layers, and an output layer, with each neuron connected to every neuron in the next layer. While FNNs are conceptually simple, they treat inputs independently and fail to capture spatial structure in image data, which is essential for understanding visual data.

Convolutional neural networks overcome this limitation by using convolutional filters that detect spatial features such as edges, textures, and patterns, making them more effective for image-related tasks.

1.2.1. Convolutional neural networks

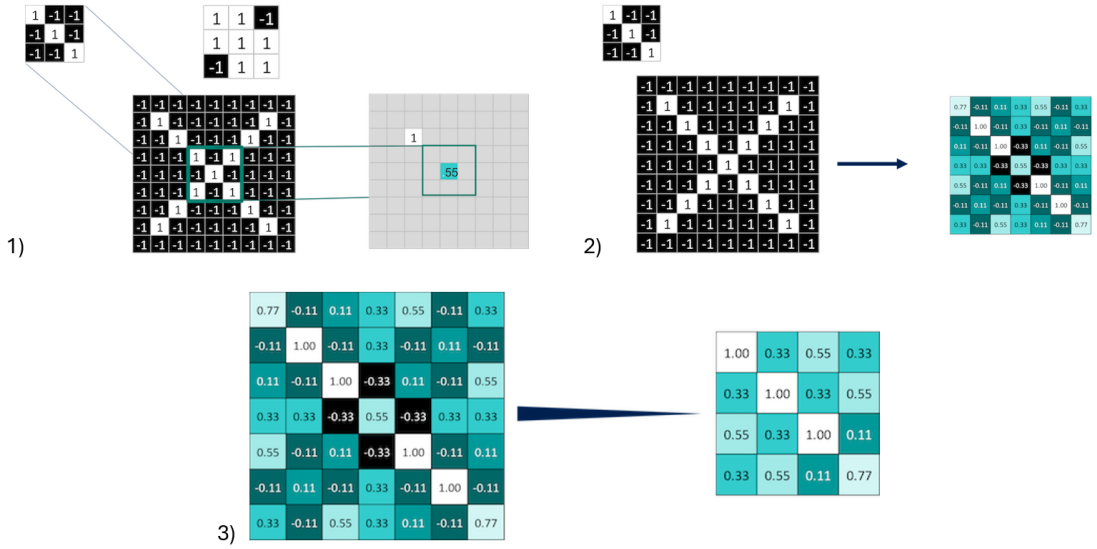


Figure 1.4: A visual demonstration of convolution and pooling operations in a convolutional neural network (CNN). (1) A kernel is convolved over a 2D image to generate a feature map by computing local dot products. (2) This process highlights features such as edges or texture patterns. (3) A pooling layer, here using 2×2 max pooling, downsamples the feature map by selecting dominant values within each region, reducing spatial dimensions while retaining key information. Adapted from [6].

Convolutional neural networks (CNNs) are particularly effective for image recognition tasks due to their ability to extract and interpret local features with translational invariance, allowing the detection of patterns regardless of their spatial position or minor distortions. CNNs achieve this through a sequence of convolutional and pooling layers that progressively transform the input data into higher-level feature representations while reducing dimensionality.

In convolutional layers, filters (or kernels) slide across the input image, computing dot products with local regions to generate feature maps that emphasise specific patterns such as edges, textures, or colours (see fig. 1.4). The early layers tend to capture low-level features, while deeper layers progressively extract more abstract representations, including shapes and semantic structures. Non-linear activation functions, typically Rectified Linear Units (ReLU), are applied to the output of each convolution to enable the network to

model complex, non-linear relationships.

Pooling layers reduce the spatial resolution of feature maps, improving computational efficiency and enhancing robustness to small spatial variations. The most common method, `MaxPooling`, retains the most prominent features by selecting the maximum value within local regions of the feature map.

By stacking multiple convolutional and pooling layers, CNNs construct a hierarchical representation of the input, enabling effective generalisation across a wide range of image recognition tasks. The final layers condense abstract features, improving classification accuracy while mitigating overfitting, thus making CNNs a powerful architecture for large-scale visual data analysis.

Training any type of neural network generally requires a large amount of training and testing data. For supervised learning, this typically means large quantities of labelled data. A notable example is the ImageNet Large Scale Visual Recognition Challenge dataset, which contains over 14.2 million annotated images [7], [8]. In supervised learning, labelled data is used to train a model to predict the labels of previously unseen samples to minimise the difference between predicted and actual labels.

However, annotating datasets, particularly those related to specialised scientific images, is a time-consuming and labour-intensive task. This presents a significant challenge in applying supervised learning to STM image datasets, where the sheer volume of data makes manual labelling infeasible.

To overcome this limitation, we turn to unsupervised machine learning, a branch of machine learning that does not require labelled data. Instead of relying on explicit supervision, unsupervised learning techniques enable the model to autonomously identify patterns, relationships, and structures within unlabelled data. This approach allows the model to learn from the data itself, discovering intrinsic groupings or anomalies that may otherwise go unnoticed. Using unsupervised learning, we can explore large datasets of STM images without the need for manual labelling, enabling the automated extraction of meaningful patterns and facilitating a more efficient analysis process.

In the context of STM images, unsupervised learning techniques such as clustering, anomaly detection, and dimensionality reduction offer powerful tools for identifying and classifying

atomic-scale features. These methods help uncover hidden structures in the data, such as adsorbed atom groups or surface defects, which would be difficult or impossible to identify manually without extensive domain knowledge.

1.2.2. Autoencoders

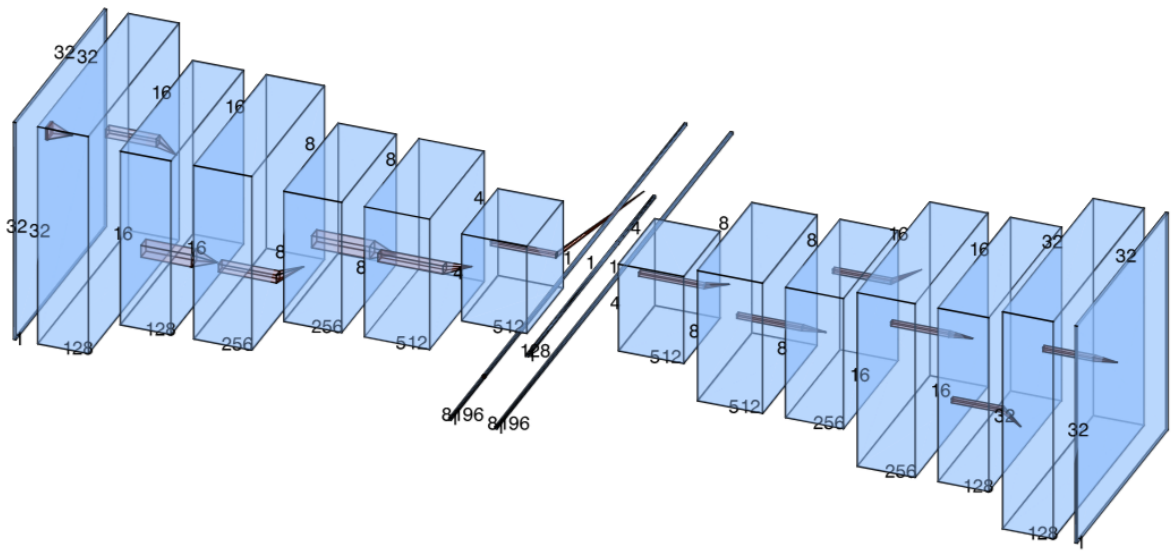


Figure 1.5: Diagram of the convolutional autoencoder architecture used in this project. The encoder progressively compresses 32×32 STM image patches through stacked convolutional and pooling layers, producing a 128-dimensional latent vector. The decoder then reconstructs the input from this compressed representation using upsampling and convolutional layers. This structure enables the network to learn efficient, spatially-aware representations of local atomic environments. Image generated using Lenail's neural network generator [9].

An autoencoder (AE) is a type of neural network commonly used for unsupervised learning, designed to learn efficient representations of data. AEs are particularly useful for tasks such as compression, noise reduction, anomaly detection, and dimensionality reduction. They excel at uncovering the underlying patterns of data without requiring labelled datasets, making them a powerful tool for analysing complex, high-dimensional data. Unlike classification networks like CNNs, which produce a label for a given input, autoencoders aim to reconstruct the input data with minimal loss of information.

Autoencoders consist of three main components: the encoder, the bottleneck layer, and the decoder (see fig. 1.5). The encoder, which operates similarly to a CNN, compresses the input image into a more compact and abstract form through a series of convolution and pooling layers. The encoder learns to capture relevant features from the data and

transforms the image into a lower-dimensional representation. This compressed form is passed through the bottleneck layer, which contains only a small number of neurons. The bottleneck layer forces the encoder to retain only the most important features of the input data, discarding any unnecessary or redundant information. This stage produces a latent vector, a reduced representation that encapsulates the core features of the original data.

The decoder then reconstructs the original image from this latent vector using up-sampling and convolution layers. The goal of the autoencoder is to minimise the difference between the input and the output of the decoder, ensuring that the reconstruction is as close as possible to the original data. Through training, the autoencoder learns to effectively encode data into a compact latent representation and then decode them back to their original form. Once trained on a large unlabelled dataset, the autoencoder can process new unseen images and generate their corresponding latent vectors. These lower-dimensional latent vectors provide a more efficient representation of the data, which can be leveraged for tasks like clustering or anomaly detection.

In the context of STM image analysis, the encoder component of an autoencoder is employed to extract latent representations from image patches. These latent vectors serve as high-level feature descriptors of the underlying atomic structures. To identify and group distinct atomic configurations, unsupervised clustering algorithms such as K-means or DBSCAN can be applied to the latent space. By grouping similar feature embeddings, the clustering process enables the automated categorisation of structurally similar regions, facilitating analysis without the need for labelled data.

Autoencoders thus provide a compact and informative representation of STM images, reducing the dimensionality of the dataset and enhancing the efficiency of downstream tasks such as clustering and classification. This approach enables the scalable and systematic analysis of large volumes of unlabelled STM data, supporting the rapid and accurate identification of recurring atomic patterns that may be difficult to detect through manual inspection.

1.3. Clustering

Clustering is a fundamental technique in unsupervised learning used to identify patterns and group similar data points without predefined labels. It is particularly useful for large datasets where manual labelling is impractical.

1.3.1. K-means

K-means is widely used for its efficiency and scalability in high-dimensional data. It partitions a dataset $X = \{x_1, x_2, \ldots, x_n\}$ into k clusters $C = \{C_1, C_2, \ldots, C_k\}$ by minimising the within-cluster sum of squares:

$$SSE(X, C_k) = \sum_{x \in X} \min |x - \mu_i|^2,$$

where μ_i is the centroid of cluster C_i. The algorithm iteratively assigns points to the nearest centroid and updates centroids until convergence.

Choosing k greatly affects performance. Common heuristics include the elbow method and silhouette score:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

where $a(i)$ is the mean intra-cluster distance, and $b(i)$ the mean distance to the nearest neighbouring cluster. Scores near 1 indicate well-separated clusters.

K-means assumes spherical clusters of similar size and density, which may not hold for STM data. It is also sensitive to centroid initialisation, so techniques like K-means++ are typically used. Despite these drawbacks, its speed and interpretability make it a useful baseline.

1.3.2. DBSCAN

Density-based clustering methods, such as DBSCAN, offer a flexible alternative to centroid-based approaches like K-means [10]. One of DBSCAN's key advantages is that it does not require the number of clusters to be specified in advance. Instead, it identifies clusters as contiguous regions of high point density, separated by areas of lower density. This makes DBSCAN particularly effective at discovering clusters of arbitrary shape and size, which is advantageous when dealing with complex, non-linear data distributions such as those found in STM images.

DBSCAN defines three types of points based on local density: *core points*, which have at least a minimum number of neighbours within a given radius; *border points*, which are within the neighbourhood of a core point but do not have enough neighbours themselves; and *noise points*, which are neither core nor border points. The algorithm requires two parameters: the neighbourhood radius ϵ, and the minimum number of points MinPts required to form a dense region. A cluster is formed by starting from a core point and iteratively expanding to include all points density-reachable from it, resulting in clusters that can take on arbitrary geometries and sizes.

Unlike K-means, DBSCAN is robust to noise and outliers, as it explicitly identifies sparse regions of the data as noise. This is particularly valuable in STM image analysis, where atomic-scale defects, vacancies, and surface irregularities are not uniformly distributed and may not conform to well-defined shapes. DBSCAN's ability to detect these anomalies without requiring manual annotation makes it well-suited to automated materials characterisation workflows.

However, the algorithm's performance is sensitive to the choice of ϵ and MinPts. Selecting values that are too small can lead to over-fragmentation, while overly large values may cause distinct clusters to be merged. In practice, tuning these parameters requires an understanding of the scale and density distribution of the data, which may not always be available. Overall, DBSCAN's flexibility and noise tolerance make it a strong candidate for clustering tasks in STM image analysis, particularly in contexts where feature boundaries are irregular and data contains significant heterogeneity or noise.

1.3.3. HDBSCAN

HDBSCAN extends DBSCAN by creating a hierarchy of clusters across multiple density levels [11]. It removes the need to set ϵ, making it more robust to heterogeneous densities and scales. While more flexible, it still struggles with the overlapping clusters or smooth feature transitions common to STM defect data.

1.3.4. Gaussian Mixture Models

Gaussian Mixture Models (GMMs) offer a fundamentally different approach to clustering by modelling the data as a weighted combination of multiple multivariate Gaussian distributions [12]. Each component in the mixture corresponds to a potential cluster and is defined by its mean vector and covariance matrix, allowing GMMs to represent overlapping group structures in high-dimensional space. Unlike hard clustering methods such as K-means, which assign each data point to a single cluster, GMMs provide soft probabilistic assignments. This means that each point is given a likelihood of belonging to each cluster, facilitating more nuanced interpretations of ambiguous data.

This soft clustering framework is particularly advantageous in contexts like STM image analysis, where defect boundaries are often blurred, and morphological distinctions between atomic features can be subtle or continuous rather than discrete. For example, adsorbate structures may gradually transition from one geometric configuration to another, and a hard boundary imposed by K-means could obscure such relationships. GMMs are better equipped to model these uncertainties, capturing overlapping or transitional defect morphologies more gracefully.

This makes GMMs well-suited for STM data, where defects often form continuous morphologies. Unlike K-means, GMMs can model ellipsoidal clusters and adapt to anisotropic latent spaces. Their flexibility and probabilistic nature enable more nuanced classification, though they do require selecting the number of components and covariance type—typically guided by AIC or BIC.

1.4. Literature Review

Clustering methods have been effectively utilised in various image analysis tasks, including biomedical imaging and materials science. In Jamal et al. [13], Principal Component Analysis (PCA) was first used to reduce the dimensionality of breast cancer imaging data, followed by K-means clustering to group the images into benign and malignant cases. This approach enabled the identification of significant patterns within the data, achieving a precision of 85.7% and a recall of 93.75%, highlighting the potential of unsupervised learning methods for diagnostic applications. Similarly, Wang et al. [14] applied K-means clustering to the analysis of transmission electron microscopy (TEM) images of metal nanoparticles. The method performed effectively due to the relatively spherical and Gaussian-distributed nature of the clusters, demonstrating K-means' suitability for structurally uniform datasets.

In the context of STM image analysis, Ziatdinov et al. [15, 16, 17] developed a deep learning framework that integrated convolutional neural networks (CNNs) with clustering techniques to identify atomic species and surface defects. Their method employed multiple CNNs to extract feature embeddings from STM data, which were then clustered using a mean-shift density-based clustering algorithm related to DBSCAN. This approach allowed for the identification of both regular atomic structures and irregular defects without requiring extensive manual labelling. While effective, their reliance on simulated or partially labelled datasets introduced challenges in scalability and generalisability, which this project aims to address by leveraging autoencoder-derived latent representations combined with fully unsupervised clustering.

Together, these studies demonstrate the value of combining dimensionality reduction with clustering techniques for the automated analysis of high-resolution microscopy data. Whether through centroid-based methods like K-means or density-based approaches such as mean-shift, these techniques provide a foundation for scalable and interpretable image segmentation tasks across different scientific domains.

2. Methodology

We implement a two-stage unsupervised segmentation pipeline for STM image analysis inspired by Ziatdinov *et al.* [15, 16, 17]. Unlike prior approaches that rely on simulated or partially labelled data, our method uses entirely unlabelled STM images to achieve automated structural characterisation with minimal supervision. This framework combines convolutional autoencoders [6] and clustering algorithms [18, 19] to segment STM images based on learned representations. Similar methods have succeeded in domains such as nanoparticle detection [14], satellite segmentation [20], and atomic defect tracking [21]. Our contribution extends these ideas to STM images by using autoencoder-derived features and probabilistic clustering to classify defects at the atomic scale.

The pipeline begins by training a convolutional autoencoder (AE1) on STM image patches, compressing each into a 32-dimensional latent vector. The encoder is then used to generate embeddings for a full STM image, which are clustered using K-means to form a coarse character map. Simultaneously, a reconstruction error map is generated by comparing input patches with their autoencoder outputs. These two maps are combined into a binary mask highlighting candidate defect regions.

These masked regions are extracted and used to train a second, more compact autoencoder (AE2) with an 8-dimensional latent space, enabling fine-grained clustering. The resulting latent embeddings are clustered using a Gaussian Mixture Model (GMM) to assign structural classes to each candidate defect. This produces a spatially resolved segmentation of atomic-scale features, allowing full STM images to be characterised without manual annotation.

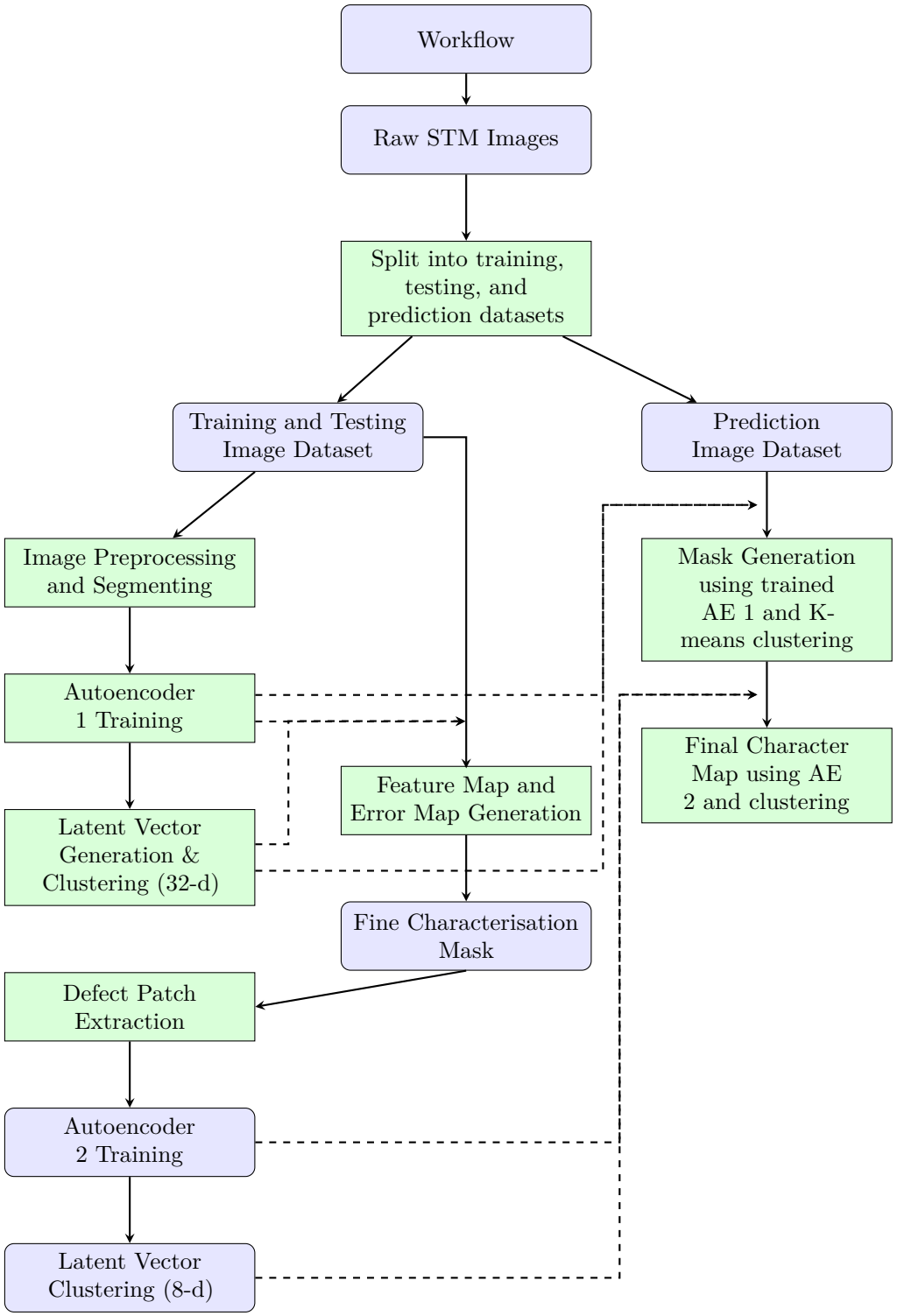


Figure 2.1: High-level overview of the two-stage STM image segmentation pipeline. The left branch shows the training pipeline for both autoencoders: AE1 is trained on general STM patches and contributes to generating the characterisation mask, while AE2 is trained only on extracted defect patches. The central column shows how feature maps and error maps are used to build a fine-grained binary characterisation mask. The right branch represents the characterisation pipeline: a new STM image is segmented using the pre-trained models. **Dashed arrows** indicate where pre-trained model components (autoencoder weights and cluster assignments) are reused during either map generation or final segmentation.

sectionData Preprocessing and Patch Extraction

Raw STM images are converted to greyscale, normalised to $[0, 1]$, and split into training, testing, and prediction sets. Each image is divided into non-overlapping 32×32 pixel patches (see fig. 2.2)—a size selected to preserve local atomic structure while avoiding excessive noise or loss of context. Smaller patches risk omitting critical spatial detail; larger ones may blend overlapping features.

Approximately 1800 STM images were processed, yielding over 300,000 patches. The dataset was split into 1450 training images, 300 testing images, and 50 for prediction. All models were trained and evaluated on UCL's LCN APOLLO high-performance cluster using NVIDIA GPUs.

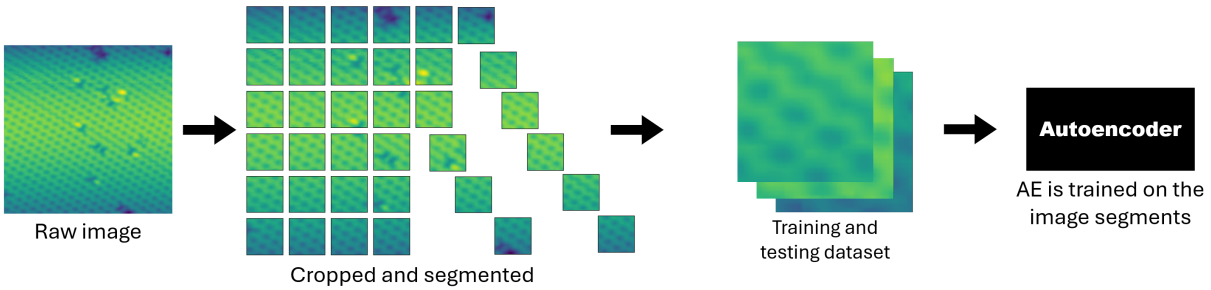


Figure 2.2: Visual overview of the data preparation process. Raw STM images are cropped and segmented into non-overlapping 32×32 patches, which are then compiled into training and testing datasets. These image segments form the input to the autoencoder for unsupervised feature learning.

2.1. Initial Feature Learning

2.1.1. Autoencoder Architecture

The first autoencoder is trained on the pre-processed image segments to extract meaningful feature representations. Its encoder comprises three convolutional layers, progressively increasing the number of filters from 128 to 512, interleaved with pooling layers to reduce spatial dimensionality (see fig. 1.5 and appendix A.1). The resulting feature maps are compressed into a 32-dimensional latent representation that captures the essential structure of the STM patches.

The decoder mirrors the encoder's architecture, using upsampling and convolutional layers to reconstruct the original input from the latent vector. A final convolutional layer with a sigmoid activation function ensures that the output intensities remain within the normalised range $[0, 1]$.

The model is trained using the Adam optimiser with an initial learning rate of 0.001, and reconstruction accuracy is measured using mean squared error (MSE) loss. To improve generalisation and accelerate convergence, early stopping and learning rate reduction techniques are applied. Additionally, batch normalisation and ReLU activations are used after each layer to stabilise training and mitigate overfitting.

2.2. General Image Segmentation

2.2.1. Latent Space Clustering

Once trained, the encoder is used to embed STM image patches into the 32-dimensional latent space. These latent vectors are then clustered using the K-means algorithm, which partitions the latent space into k distinct groups—each representing a structurally similar class of STM patches. Prior to clustering, MinMax normalisation is applied to ensure uniform weighting across all feature dimensions.

K-means was chosen for its simplicity, scalability, and ease of interpretation in high-dimensional settings—particularly when the number of clusters can be estimated using properties such as the elbow method or silhouette analysis.

Alternative clustering techniques were evaluated but ultimately discarded. DBSCAN and HDBSCAN exhibited high sensitivity to density parameters and consistently failed to resolve uniformly distributed clusters within the latent space. Agglomerative clustering, while more flexible in principle, was found to be computationally infeasible due to the large number of latent vectors (over 300,000).

While effective in identifying broad structural groupings, the use of K-means clustering assumes isotropic, spherical feature distributions in latent space. This may limit the resolution of more subtle or rotation-sensitive defects. Moreover, reconstruction error alone may fail to identify defects with low contrast or highly localised distortions. Future iterations could incorporate contrastive learning or physically motivated priors to improve clustering fidelity.

The resulting cluster assignments are mapped back onto the spatial positions of the original patches, producing a characterisation map that groups STM regions by structural similarity.

This map, while coarse, provides the first layer of defect detection by highlighting spatial heterogeneity and atomic pattern variations.

2.2.2. Character Map Generation

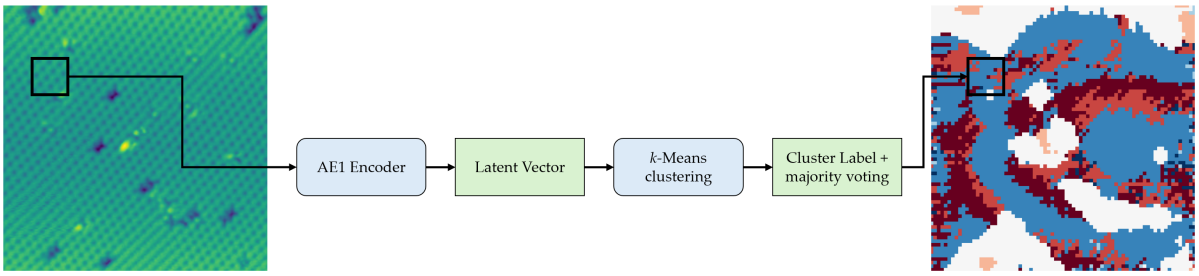


Figure 2.3: Overview of the structural segmentation pipeline. A sliding window extracts local STM patches, which are embedded into a 32-dimensional latent space using the encoder of Autoencoder 1 (AE1). Latent vectors are clustered using k-means, and majority voting is used to assign each pixel a cluster label. Notably, this process ignores spatial continuity or physics-informed constraints, which can lead to noisy cluster boundaries in regions with overlapping atomic features.

Two complementary approaches were investigated for the characterisation of images using the trained autoencoder model: (i) the generation of feature-based clustering maps and (ii) the construction of reconstruction error maps. Each method provides distinct but related insights into the underlying atomic structures present in the STM data, aimed at identifying both typical surface features and atomic-scale defects.

The first method generates a structural feature map by scanning the STM image with a sliding window of size 32×32 pixels. A stride of $s \approx 4{-}8$ pixels is used, depending on the overall image size. Unlike non-overlapping tiling, this overlapping window approach is essential for producing a dense, pixel-level characterisation of the image. Each windowed patch is encoded into a latent vector using the trained autoencoder, and this vector is subsequently clustered via the K-means algorithm to assign a label.

Because each pixel is typically covered by multiple overlapping windows, it may receive several cluster label assignments. To assign a final label per pixel, majority voting is applied over the cluster labels of all patches containing that pixel. This not only improves spatial resolution but also enhances label consistency, suppresses local outliers, and yields smoother, more coherent character maps.

2.2.3. Reconstruction Error Mapping

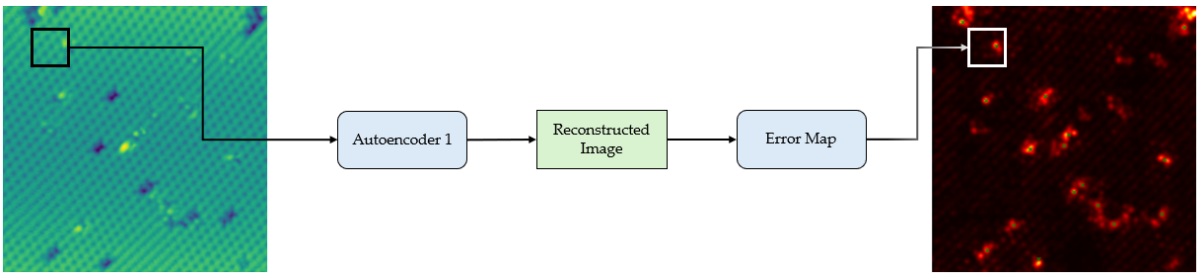


Figure 2.4: Reconstruction-based anomaly detection. Local STM patches are passed through the first autoencoder to produce a reconstructed image. The pixel-wise absolute difference between input and output yields an error map, which highlights anomalous or non-repeating atomic features. This step is sensitive to features under-represented in the training distribution and tends to produce higher errors near STM image edges.

The second method generates a reconstruction error map to identify potential structural anomalies in the STM image. As in the clustering-based feature map, the image is partitioned into overlapping 32×32 patches using a sliding window with a stride of 16 pixels. Each patch is passed through the entire autoencoder, which attempts to reconstruct it. The reconstruction error is computed as the element-wise squared difference between the input patch and its reconstruction, capturing deviations from the learned manifold of typical surface structures.

To translate these patch-level errors into a full-image representation, the reconstruction errors are aggregated across the image. For each pixel, the mean squared error is averaged over the number of windows in which it appears, producing a dense and continuous error map. Regions with high reconstruction error indicate areas where the model was unable to accurately represent the input—typically due to rare or defective atomic configurations not seen during training. To improve robustness, the reconstruction error map is smoothed using a Gaussian filter, which suppresses pixel-level noise and accentuates spatially coherent anomalies.

2.3. Defect Mask Construction

2.3.1. Combining Feature and Error Maps

A binary defect mask is generated by combining the reconstruction error map and the character map, followed by a series of morphological operations to suppress noise and

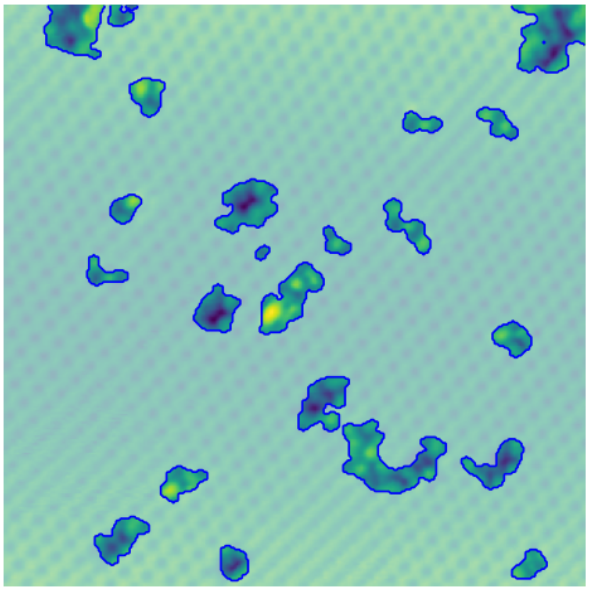


Figure 2.5: Binary mask obtained by thresholding the reconstruction error map. This highlights regions of high reconstruction loss corresponding to structural defects or adsorbates. Although effective in localising high-contrast features, the mask can miss more subtle surface variations or blur features near patch boundaries.

enhance blob-like features. This mask serves as a coarse segmentation of defect regions, preparing them for more fine-grained analysis.

The character map contributes structural and categorical information by segmenting the image into distinct atomic groups, while the reconstruction error map highlights outliers and anomalies. These maps are combined by taking the logical union (a pixel-wise OR operation) of the high-error regions and specific target clusters identified within the character map. The result is a binary characterisation mask that encodes both geometrical irregularities and structural abnormalities.

To improve segmentation quality, the raw characterisation mask undergoes a sequence of morphological post-processing steps. Initially, line-like artefacts are removed based on aspect ratio filtering. This is followed by erosion to eliminate small protrusions, Gaussian smoothing to promote blob-like consistency, and dilation to recover eroded regions. Small regions are further filtered out based on area thresholds, and morphological opening is applied to smooth the boundaries of the remaining regions. The final output is a compact, noise-reduced mask that delineates potential defect zones suitable for downstream clustering.

2.4. Defect-Specific Feature Learning

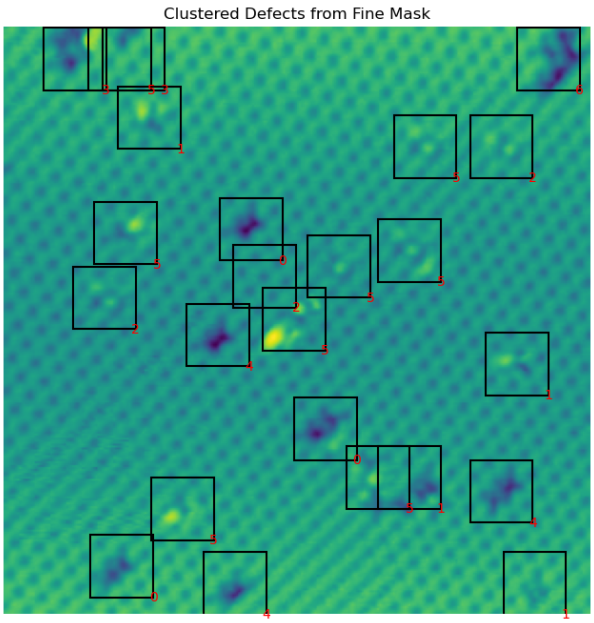


Figure 2.6: Penultimate stage of the defect classification pipeline. Bounding boxes are drawn around selected regions from the binary mask, and their corresponding patches are passed through a second compact autoencoder.

After constructing the binary defect mask, image regions suspected to contain defects are systematically extracted for further analysis. Specifically, all overlapping 32×32 image patches whose centres fall within the binary defect mask are selected and compiled into a secondary dataset. This ensures a thorough sampling of candidate defect regions while minimising redundancy.

By isolating only structurally abnormal areas, this targeted sampling strategy enables the second-stage autoencoder to focus exclusively on rare or defective features. As a result, the model can learn a more compact and discriminative representations of defect patterns without being influenced by common atomic arrangements. The architecture of this secondary autoencoder and its associated clustering method are described in the next section.

2.4.1. Secondary Autoencoder

To better distinguish between defect types, a second, more compact autoencoder is trained exclusively on the subset of image patches extracted from the binary defect mask. This

targeted training strategy avoids the problem of *feature dilution*, which occurs when a single model is exposed to both dominant atomic patterns and infrequent defect types. In such cases, the model may over-fit to the majority class, failing to learn useful representations of the minority structures [22]. By contrast, the second-stage autoencoder is explicitly designed to isolate and encode only the most structurally deviant regions.

The secondary autoencoder adopts a narrower latent space of 8 neurons to produce a more discriminative encoding of the input. Its architecture mirrors the structure of the initial autoencoder but employs reduced filter sizes and fewer parameters, making it computationally lighter and better suited for modelling local structural anomalies (see appendix A.2). Training follows the same procedure—using the Adam optimiser with a learning rate of 0.001 and mean squared error loss—along with early stopping and learning rate reduction to ensure convergence and generalisability. The training behaviour and reconstruction outputs of the defect-specific autoencoder are shown in fig. A.2 and fig. A.3, respectively.

Prior to training, each input patch is normalised between $0-1$ to ensure consistent intensity scaling across the dataset and to enhance the model's sensitivity to subtle variations in defect morphology.

Once trained, the encoder is used to project all extracted defect patches into 8-dimension latent space. These are then clustered using unsupervised algorithms to produce a refined segmentation of the defect regions, enabling the distinction of multiple defect classes based solely on structural similarity.

2.5. Clustering of Defect Patches

2.5.1. Clustering Methods

Once the defect patches are encoded by the secondary autoencoder, the resulting latent vectors are clustered to uncover structurally distinct defect classes in a fully unsupervised manner. As DBSCAN was previously investigated and found to be overly sensitive to parameter choices—often failing to capture fine-grained or uniformly distributed structures—it was not considered suitable for this stage of analysis. We focused here on

two more flexible alternatives: HDBSCAN and Gaussian Mixture Models (GMMs). Gaussian Mixture Models ultimately provided the most stable and interpretable results.

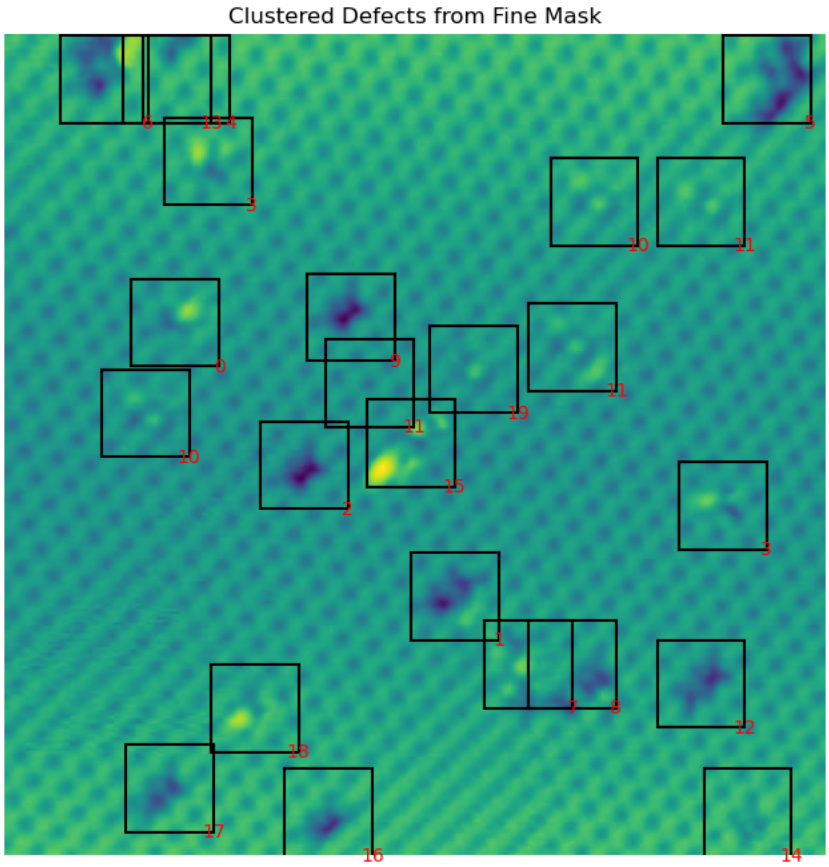
2.6. From Patch-Level Clustering to Full Image Analysis

With the defect clustering pipeline established, the final step involves applying the trained models to the full STM images to generate interpretable characterisations (fig. 2.1). Each image is systematically divided into overlapping 32×32 patches, which are passed through the primary autoencoder to obtain both latent vectors and reconstruction errors. These are used to construct the initial coarse characterisation mask by combining high-error regions with clustered atomic groups from the latent space.

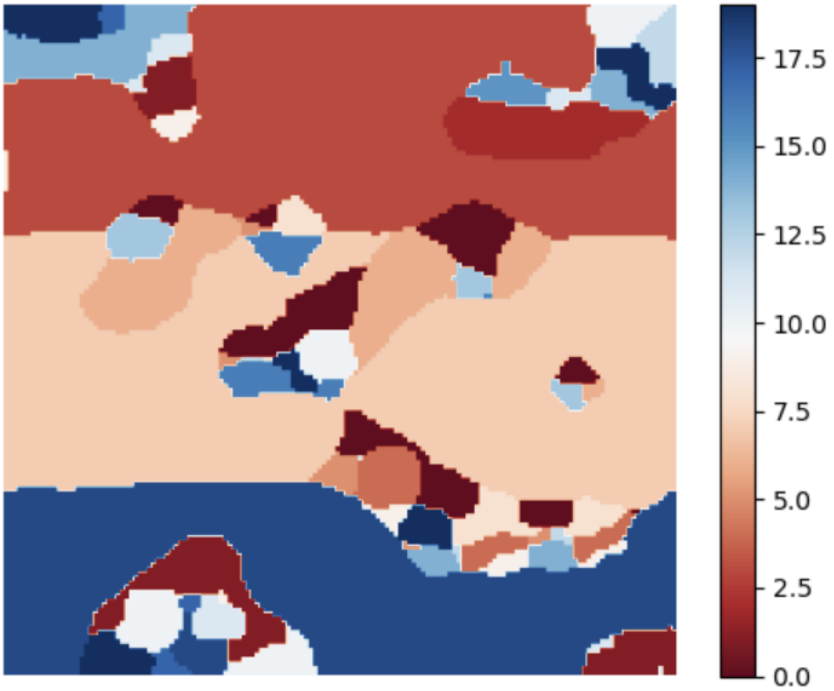
Next, the refined binary defect mask is used to extract defect-rich regions from each STM image. Overlapping 32×32 patches whose centres fall within these regions are passed through the secondary autoencoder to produce low-dimensional latent embeddings. These embeddings are then clustered using the trained Gaussian Mixture Model, assigning each candidate defect to a specific structural class.

The result of this process is a spatially resolved classification of defects across the full image. As shown in fig. 2.7, this output can be visualised in two complementary ways. Panel (a) displays bounding boxes around high-error regions, each with its assigned GMM label, offering localised and interpretable clustering of individual defect candidates. Panel (b) presents a colourised character map in which each pixel reflects the most dominant defect class across overlapping patches—enabling a coarser but more spatially continuous view of the image's structural composition.

Together, these outputs form the final stage of the pipeline, enabling fully unsupervised segmentation, classification, and visual interpretation of diverse atomic-scale features within STM images. The following section evaluates the performance and robustness of this framework using both qualitative and quantitative analyses.



(a) Clustered defects derived from the fine binary mask. Bounding boxes are drawn around high-error regions extracted from the reconstruction error map, and each patch is embedded via a secondary autoencoder into a low-dimensional latent space. Gaussian Mixture Model clustering assigns a class label (shown in red) to each defect. Although several structurally distinct defect types are identified, local label noise and class fragmentation are evident due to overlapping structures and subtle morphological variations.



(b) Alternative visualisation of the character map generated from GMM clustering of latent defect embeddings. Each colour corresponds to a distinct defect cluster, allowing coarse spatial localisation of defect types. This representation improves interpretability over bounding boxes alone, though it lacks precise structural boundaries and is sensitive to patch overlap and resolution at edges.

Figure 2.7: Final segmentation outputs from the defect classification pipeline. (a) Clustered defect bounding boxes with GMM-assigned labels. (b) Colourised character map for spatial visualisation of defect types. While bounding boxes enable localised extraction and clustering, the character map supports higher-level interpretation across the full STM image.

3. Results and Discussion

3.1. Autoencoder Performance and Latent Space

To assess the impact of latent space dimensionality on autoencoder performance, five convolutional autoencoders were trained in parallel, each with a bottleneck size of 16, 32, 64, 128, or 256 neurons. All models shared the same architecture and were trained for 150 epochs under identical conditions.

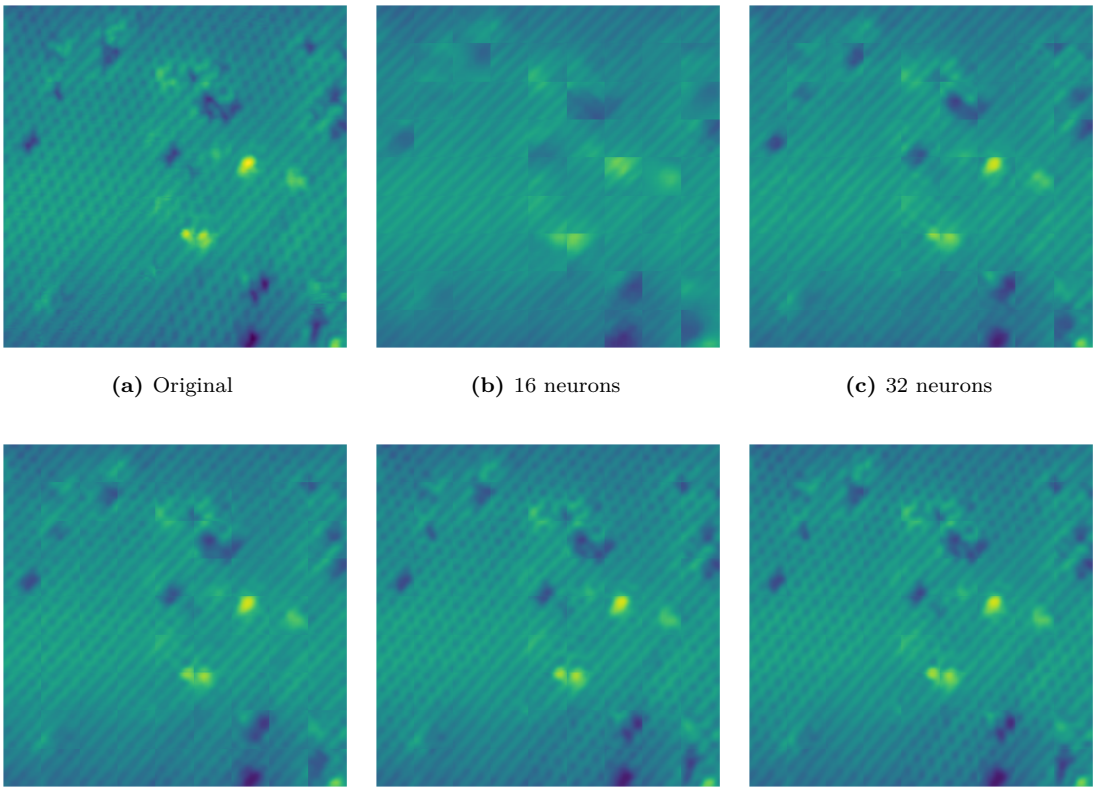
Figure A.1 and table 3.1 present the training histories and results respectively. Increasing the latent dimensionality reduced both training and validation loss, with the 256-neuron model achieving the lowest final error. However, this improvement came at a significant cost: higher dimension latent spaces increased training time and parameter count while offering diminishing visual or numerical gains.

The 16-neuron model clearly underfit the data, with unstable validation loss and reconstructions lacking fine detail. In contrast, the 32-neuron autoencoder provided a marked improvement in reconstruction fidelity, achieving stable convergence and capturing key surface features such as atomic lattices and defect contours. Although the 64-, 128-, and 256-neuron models produced progressively sharper reconstructions, the enhancements were minor relative to their increased complexity.

This trade-off is illustrated in fig. 3.1, where the 16-neuron output exhibits blocky artefacts and fails to resolve high-frequency structure. The 32-neuron model, however, reconstructs the original image with high fidelity. Beyond this point, further dimensional increases result in diminishing returns.

Based on this, the 32-neuron model was selected for downstream clustering and

segmentation tasks. Its latent space was sufficiently expressive to retain critical structural information while remaining compact enough to promote clustering. Notably, it achieved a compression/error ratio exceeding 35,000, indicating high representational efficiency with minimal dimensionality. Beyond this point, compression efficiency decreased markedly, with only marginal improvements in reconstruction error for higher-dimensional models. This choice is further supported by Bellman's work on the "Curse of Dimensionality," which states that "as the dimensionality increases, the volume of the space increases so fast that the available data become sparse" [23]. In high-dimensional latent spaces, this sparsity reduces the effectiveness of distance-based algorithms like K-means and increases the risk of overfitting due to the lack of meaningful neighbourhood structures.



(a) Original **(b)** 16 neurons **(c)** 32 neurons

(d) 64 neurons **(e)** 128 neurons **(f)** 256 neurons

Figure 3.1: Comparison of the original STM image (a) and reconstructions from autoencoders with different latent vector sizes. The 16-neuron model (b) fails to retain the lattice structure. The 32-neuron model (c) achieves a visually faithful reconstruction, while larger models (d–f) show only incremental improvements.

Table 3.1: Comparison of autoencoders with varying latent dimensions. The 32-neuron model offers a balance between reconstruction accuracy, efficiency, and model complexity. Although all models were trained for 150 epochs, the 16-neuron autoencoder exhibited a longer total training time, suggesting slower per-epoch convergence likely due to representational bottlenecks.

Latent Vector	Training Loss	Validation Loss	Compression/Error Ratio	Parameters
16	1.30×10^{-3}	1.34×10^{-3}	49000	3,226,129
32	9.12×10^{-4}	8.91×10^{-4}	35000	3,488,289
64	7.35×10^{-4}	6.76×10^{-4}	21768	4,012,611
128	4.89×10^{-4}	4.19×10^{-4}	16359	5,061,249
256	4.01×10^{-4}	3.15×10^{-4}	10000	7,158,529

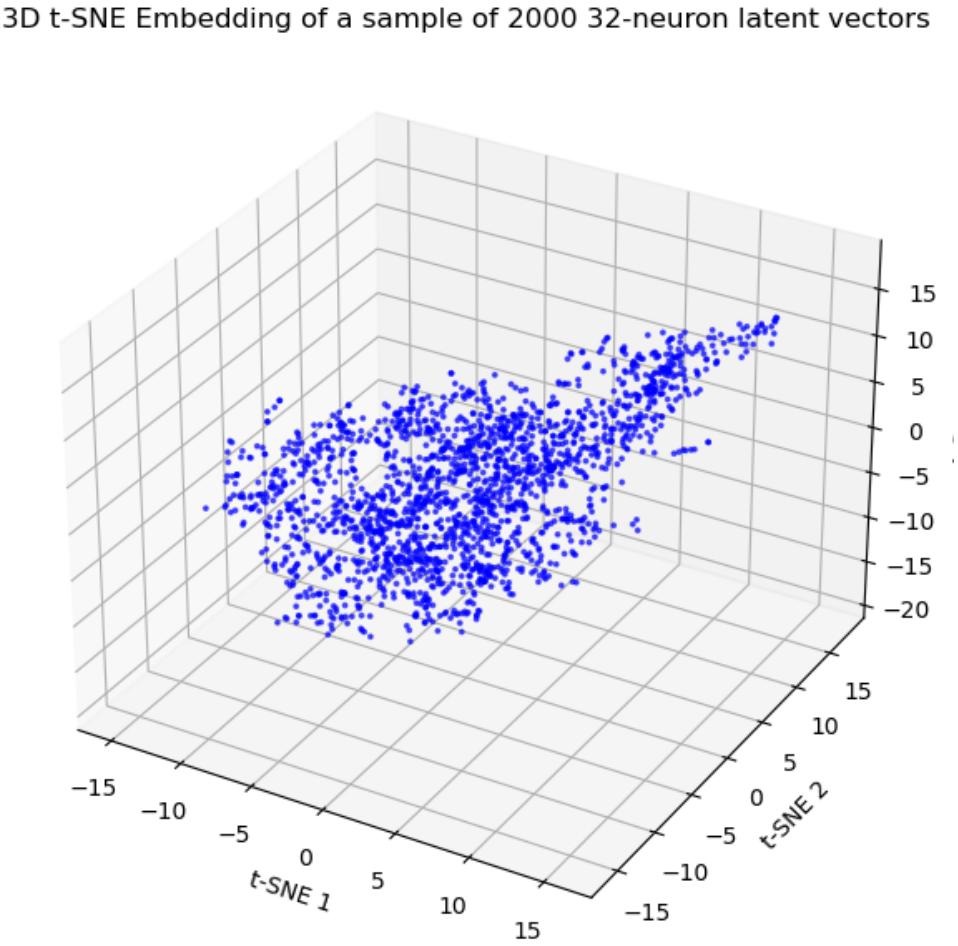


Figure 3.2: Three-dimensional t-SNE projection of 2000 randomly sampled 32-dimensional latent vectors from the primary autoencoder. The embedding qualitatively preserves local neighbourhoods in the latent space, enabling visual inspection of structural variability. However, as a non-linear dimensionality reduction method, t-SNE distorts global distances and should not be used to infer true cluster separability or density.

3.2. Clustering General Image Features

A three-dimensional t-SNE embedding of the 32-dimensional latent vectors is presented in fig. 3.2 to provide a qualitative illustration of the latent space structure. Although the t-SNE algorithm projects data into a lower-dimensional manifold using non-linear transformations, it primarily preserves local neighbourhoods rather than global distances [24]. Therefore, apparent separability in this visualisation does not necessarily correspond to true separability in the original latent space.

Nonetheless, the emergence of distinct clusters and smooth transitions in the t-SNE projection suggests that the encoder has learned to organise surface patches according to meaningful atomic-scale variations. These observations offer qualitative support for the subsequent application of unsupervised clustering algorithms. However, quantitative assessments (e.g., silhouette scores, Davies–Bouldin index) were used to robustly evaluate cluster separability in the original latent space.

3.2.1. General Clustering Model Selection

K-means was ultimately selected for the coarse-grained characterisation due to its robustness, speed, and high-quality segmentation in this context. The combination of t-SNE visualisation, elbow curve analysis, and silhouette scoring provides strong evidence for this choice.

We evaluated the number of clusters k using the elbow method and the silhouette score. As shown in fig. 3.3, the inertia curve decreases rapidly up to $k = 6$, after which further reductions become marginal, indicating diminishing returns. The first derivative plot highlights a flattening of inertia decrease beyond this point. While the silhouette score reaches its peak at $k = 2$, it declines sharply thereafter and begins to plateau in the range $k = 6$ to $k = 10$. This flattening suggests that increasing k beyond 6 offers little additional benefit in terms of cluster cohesion and separation. Based on these combined indicators, $k = 6$ was selected as a reasonable compromise between compactness and interpretability.

However there are concerns regarding the reliability of the elbow method, particularly in high-dimensional or noisy settings [25]. Specifically, it lacks a formal statistical basis, is

sensitive to scale and noise, and often yields ambiguous or misleading inflection points. Accordingly, the selected value of k is treated as a heuristic rather than a statistically validated optimum. Additional metrics such as the silhouette score were therefore employed to further assess cluster quality and ensure robustness.

The resulting cluster assignments were mapped back onto the spatial positions of the original patches to produce a coarse-grained characterisation map, grouping STM regions by structural similarity as described in the previous section.

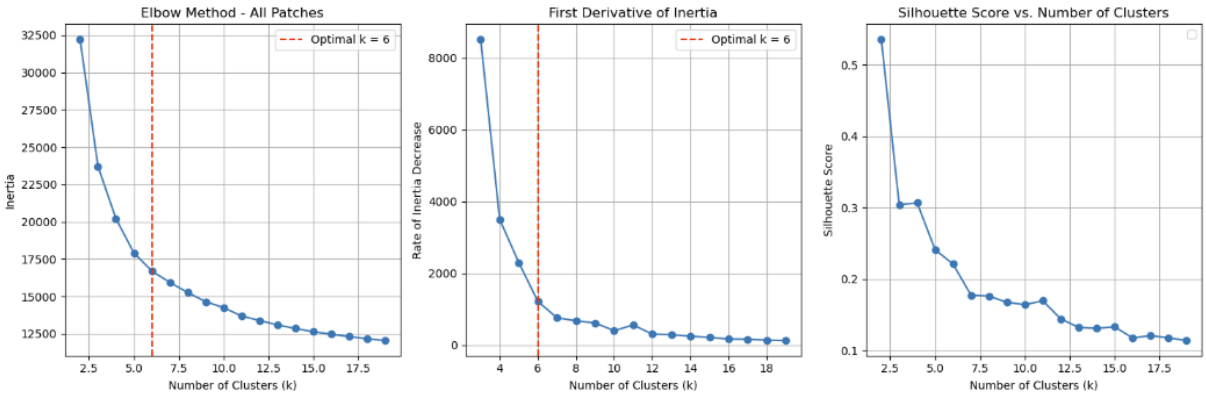


Figure 3.3: Selection of the number of clusters k using multiple K-means heuristics. **Left**: Elbow plot showing inertia vs. k, with diminishing returns beyond $k = 6$. **Centre**: First derivative of inertia, indicating a sharp decrease before flattening near $k = 6$. **Right**: Silhouette score across cluster counts, peaking at $k = 2$ but decreasing steadily thereafter. The combined metrics suggest $k = 6$ as a compromise between compactness and interpretability, though no global optimum is apparent.

3.2.1.1. Comparison with Alternative Clustering Methods

- **DBSCAN**: Though theoretically capable of discovering non-globular clusters and handling noise, DBSCAN proved highly sensitive to hyperparameters. As shown in fig. 3.4, across the ϵ and `min_samples` grid, silhouette scores remained poor, and noise rates were frequently excessive. Many settings collapsed into a single cluster, while others yielded nearly 100% noise classification. These behaviours are likely due to the mild density gradients in the high-dimensional latent space and the absence of well-defined cluster boundaries.

- **HDBSCAN**: This algorithm, an extension of DBSCAN, offers improved flexibility and soft clustering assignments. As illustrated in fig. 3.5, HDBSCAN achieved lower noise ratios and a wider variety of cluster counts. However, silhouette scores remained low or negative, suggesting fuzzy or overlapping clusters. While it occasionally

produced meaningful splits in small latent-space subsets, HDBSCAN was unstable at scale and sensitive to minor hyperparameter shifts. The number of clusters varied significantly across settings, undermining consistency and interpretability.

- **Agglomerative Clustering (Hierarchical)**: This method was considered at small scales (10,000 samples), but the memory and time complexity ($O(n^3)$) made it infeasible at full dataset scale. The dendrograms revealed several plausible segmentation points, but cluster boundaries were difficult to interpret physically.

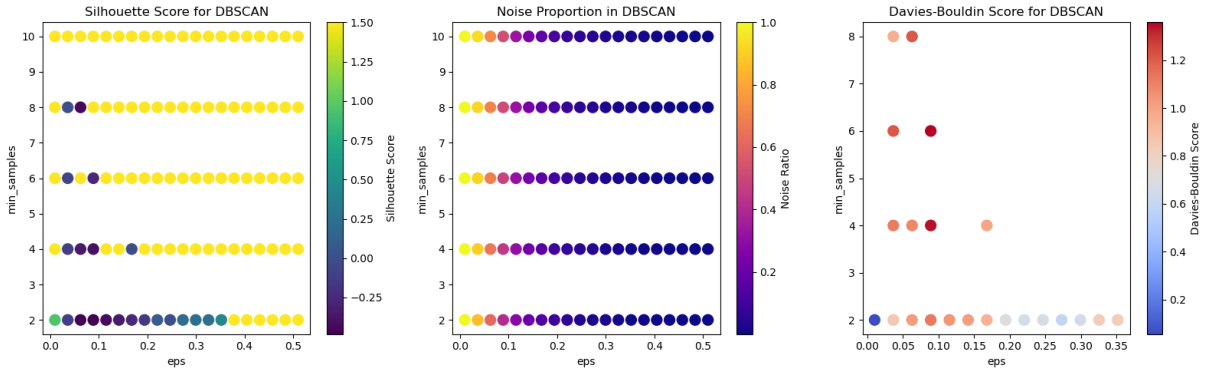


Figure 3.4: DBSCAN performance over a grid of ε and `min_samples` values. **Left**: Silhouette scores, with low values indicating poor cluster compactness and separation. **Middle**: Noise ratio, defined as the proportion of points labelled as noise; high values indicate over-sensitivity to density fluctuations. **Right**: Davies–Bouldin index scores, where lower values represent better clustering. DBSCAN consistently exhibits unstable behaviour, with many parameter combinations leading to degenerate solutions (e.g. single clusters or excessive noise).

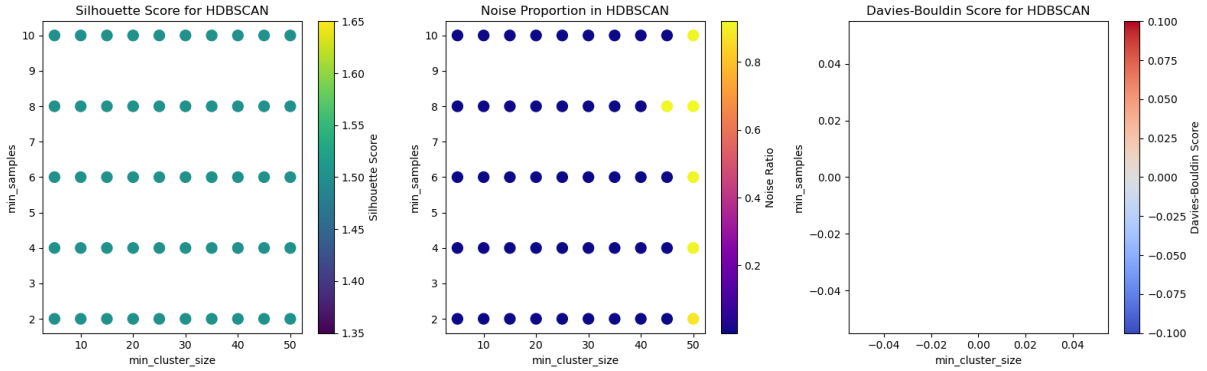


Figure 3.5: HDBSCAN performance as a function of `min_samples` and `min_cluster_size`. **Left**: Silhouette scores, mostly negative or near zero, suggesting overlapping or ambiguous cluster boundaries. **Middle**: Noise ratio, generally low, though some regions exhibit fragmentation. **Right**: Davies-Bouldin score is blank as no model could generate more than 3 clusters

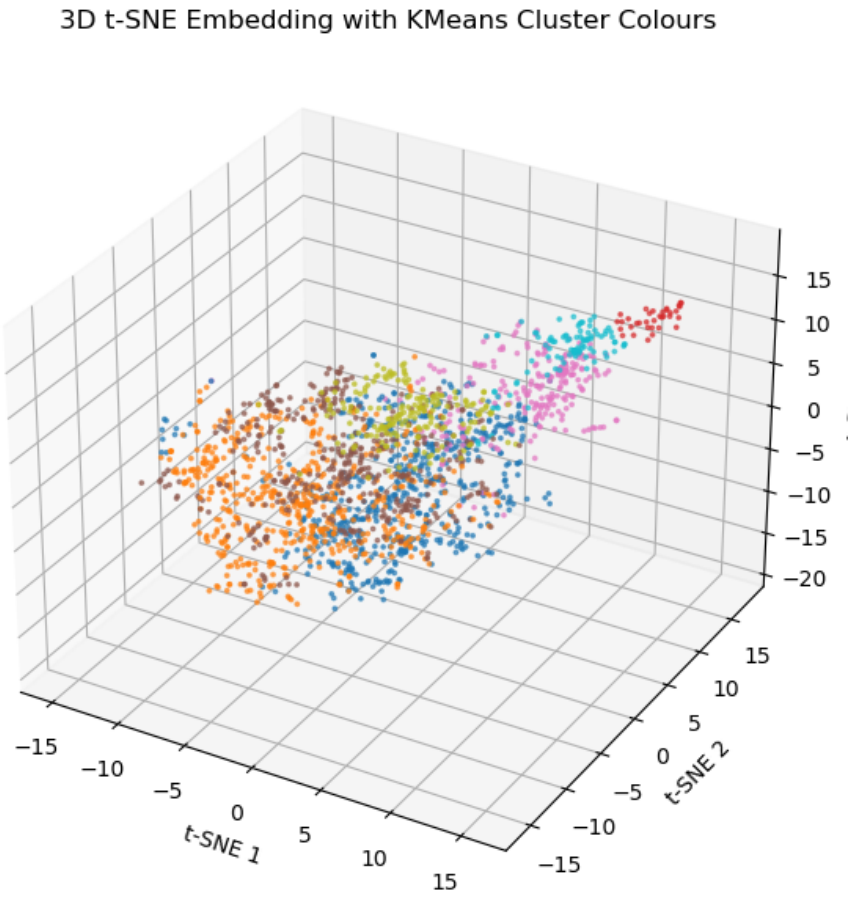


Figure 3.6: t-SNE projection coloured by K-means cluster assignments (with $k = 6$). Points are coloured by their cluster label, allowing a visual cross-check between clustering outputs and latent manifold geometry. The appearance of compact regions with dominant colours suggests that the latent space encodes meaningful atomic-scale distinctions. However, the projection may exaggerate apparent separation due to the embedding's non-linear nature.

3.3. Clustering Defect Features

Following extensive evaluation, the final clustering model selected for STM defect segmentation was a Gaussian Mixture Model (GMM) with full covariance structure and $k = 20$ components. This configuration provided the best balance between model complexity and clustering quality, as determined by Davies–Bouldin index (DBI) and Bayesian Information Criterion (BIC). The following sections outline the methodology, alternatives considered, and justification for this choice.

Following the projection of extracted defect patches into the 8-dimensional latent space using the secondary autoencoder, several clustering methods were explored to identify distinct defect categories. Due to the reduced dimensionality of the latent representation, we initially hypothesised that density-based clustering algorithms such as DBSCAN and HDBSCAN might be better suited to capture fine-grained variations in defect structure.

3.3.1. DBSCAN and HDBSCAN for Defect Clustering

In principle, lower-dimensional embeddings should improve DBSCAN's ability to detect clusters by mitigating the sparsity issues present in higher-dimensional spaces. However, as shown in fig. 3.7, DBSCAN failed to produce stable segmentations across a broad range of ϵ and `min_samples` values. The algorithm frequently collapsed all points into a single cluster or misclassified most data as noise, suggesting that the latent space lacked the sharply defined density peaks required for DBSCAN to robustly identify cluster boundaries.

To address these limitations, we investigated HDBSCAN, an extension that adaptively adjusts density thresholds and constructs a hierarchy of potential clusters. HDBSCAN yielded more stable outputs across parameter ranges and occasionally recovered plausible structural groupings. However, it often over-segmented the data into numerous small clusters or retained a high proportion of noise labels. These behaviours were particularly pronounced in regions of the latent space where defect variation was continuous rather than discrete, limiting the interpretability and consistency of the output (see fig. 3.8).

3.3.2. GMM for Defect Clustering

Given the challenges with density-based methods, we adopted Gaussian Mixture Models (GMMs) as a more flexible and probabilistically grounded alternative. GMMs model the data as a mixture of multivariate Gaussian distributions and provide soft cluster assignments, which are particularly useful in capturing the ambiguous and overlapping morphologies often seen in STM defect embeddings.

To identify the optimal configuration, we performed a grid search across four covariance types (spherical, diagonal, tied, and full) and a range of component counts ($k = 2$ to 30). Cluster quality was assessed using silhouette score, Davies–Bouldin index (DBI), and the information-theoretic criteria Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC).

Selecting the optimal model in an unsupervised context is inherently difficult due to the absence of ground truth labels. While silhouette score and DBI offer useful measures of clustering compactness and separation, they can be sensitive to soft assignments and latent noise. According to the `scikit-learn` documentation, model selection for GMMs is best guided by minimising AIC and BIC, which jointly evaluate model fit and complexity [12]. Our implementation used the built-in `gmm.bic()` and `gmm.aic()` functions, which return positively valued scores that are minimised when the model best captures the data without overfitting. In our plots, lower values indicate better models, consistent with this convention.

3.3.3. Defect Clustering Model Choice

The selected model, with $k = 20$ components and full covariance, offered sufficient flexibility to capture the nuanced structure of defect embeddings while remaining interpretable and resistant to overfitting.

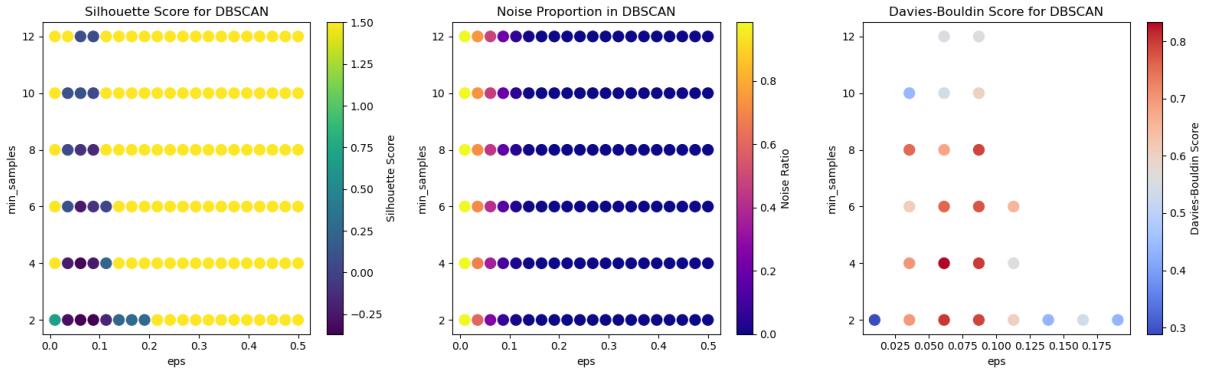


Figure 3.7: Evaluation of DBSCAN performance across ε and `min_samples` values. **Left**: Silhouette scores remain low or negative, indicating poor cluster separation. **Middle**: Noise ratio, with high values (>0.8) for most parameter combinations. **Right**: Davies–Bouldin index scores, with only slight improvements at narrow parameter ranges. The instability across all metrics highlights DBSCAN's unsuitability in this context.

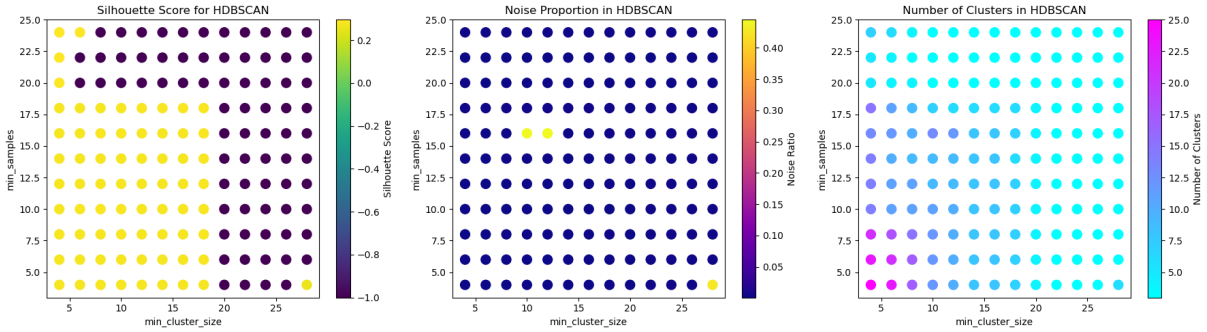


Figure 3.8: Grid search over `min_samples` and `min_cluster_size` for HDBSCAN. **Left**: Silhouette scores are consistently low or negative. **Middle**: Noise proportion remains low across most settings, suggesting improved robustness over DBSCAN. **Right**: Number of clusters, indicating high sensitivity to small changes in input parameters. Despite its flexibility, HDBSCAN exhibited inconsistent clustering performance in this latent space.

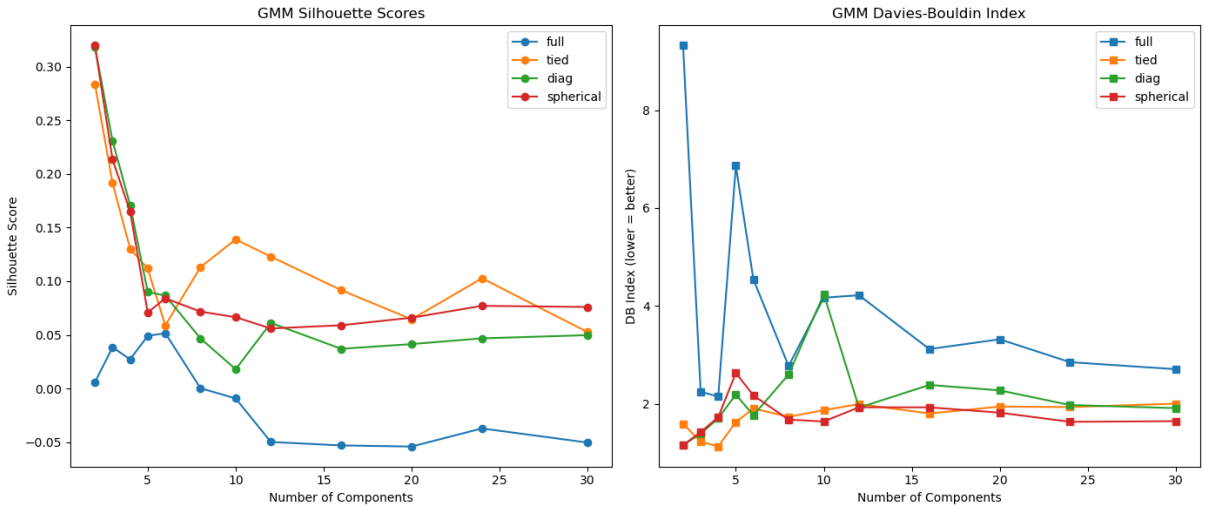


Figure 3.9: Gaussian Mixture Model (GMM) evaluation across different numbers of components and covariance types. **Left**: Silhouette scores for each configuration; 'spherical' and 'tied' covariance options performed best at low component counts. **Right**: Davies–Bouldin index shows lowest values for GMMs with 5–7 components and diagonal or spherical covariance matrices, indicating compact and well-separated clusters. These metrics guided the selection of a 6-component GMM with spherical covariance.

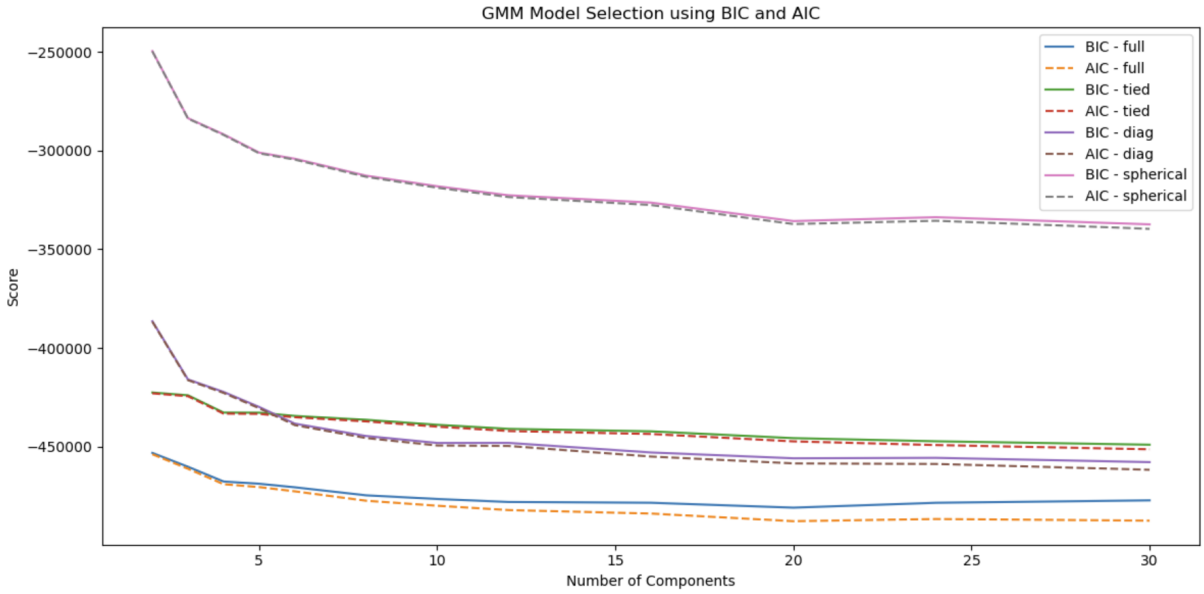


Figure 3.10: Model selection criteria for GMMs using AIC and BIC across component counts and covariance types. The full covariance model achieves the lowest Bayesian Information Criterion (BIC) score of -480943 at $k = 20$, indicating the best overall balance between model fit and complexity.

3.4. Limitations

While the proposed two-stage pipeline demonstrated robust performance across a variety of STM images—including regions near image boundaries—it exhibited notable limitations when applied to areas containing atomic step edges or large-scale structural gradients. In such cases, the latent feature space struggled to distinguish between true atomic defects and extended contrast transitions, resulting in excessive false positives or inconsistent cluster assignments.

This issue is most apparent in the final stage of the pipeline, where defect detection is guided by localised reconstruction error and unsupervised clustering. Step edges, which naturally present abrupt intensity shifts and topographic discontinuities, can produce high reconstruction loss even in the absence of genuine defects. Although we attempted to mitigate this by training the second autoencoder on normalised defect patches—thereby boosting local structural contrast within each patch—these high-gradient regions often dominated the fine characterisation mask and led to over-segmentation.

To counter this, we applied a series of morphological operations to the binary defect mask. These steps were specifically designed to suppress elongated or repetitive line-like artefacts that often arise near step edges. By filtering regions based on aspect ratio, performing erosion and dilation, and removing geometrically consistent linear features, we aimed to reduce spurious bounding box generation and improve the spatial coherence of the final output.

Nevertheless, the clustering model—trained primarily on compact, point-like morphologies—remained poorly suited to capturing the extended, edge-aligned structure of atomic steps. As a result, these regions often received overlapping or fragmented cluster labels, further reducing interpretability.

3.4.1. Failure Examples

Despite the pipeline's strong performance in many scenarios, certain STM features remain difficult to classify. These failure cases highlight limitations in the model's generalisability and offer insights for future improvement.

Figure 3.11 shows a case where the system correctly suppresses most of the background but mistakenly flags low-contrast lattice patterns as defects. These false positives likely arise from under-represented textures in the training set. Notably, the reconstruction error map (Figure B.9) appears inverted—visually uniform regions yield high loss, while bright surface features are reconstructed well. This suggests that the autoencoder is biased toward high-contrast training examples, leading to occasional mislabelling and inconsistent clustering.

In fig. 3.12, a diagonal atomic step edge is incorrectly flagged as defective (see fig. B.13). These extended contrast transitions produce high reconstruction loss, resulting in dense, overlapping bounding boxes with fragmented cluster labels. This reflects the limitations of relying purely on local reconstruction error, especially for continuous or elongated features like step edges.

To address these issues, future versions could incorporate classical edge detectors such as Canny or Sobel filters to pre-segment topographic gradients. These regions could then be excluded from the defect mask or processed with specialised models. Alternatively, embedding positional or gradient-aware features into the autoencoder could help distinguish real defects from benign contrast variations.

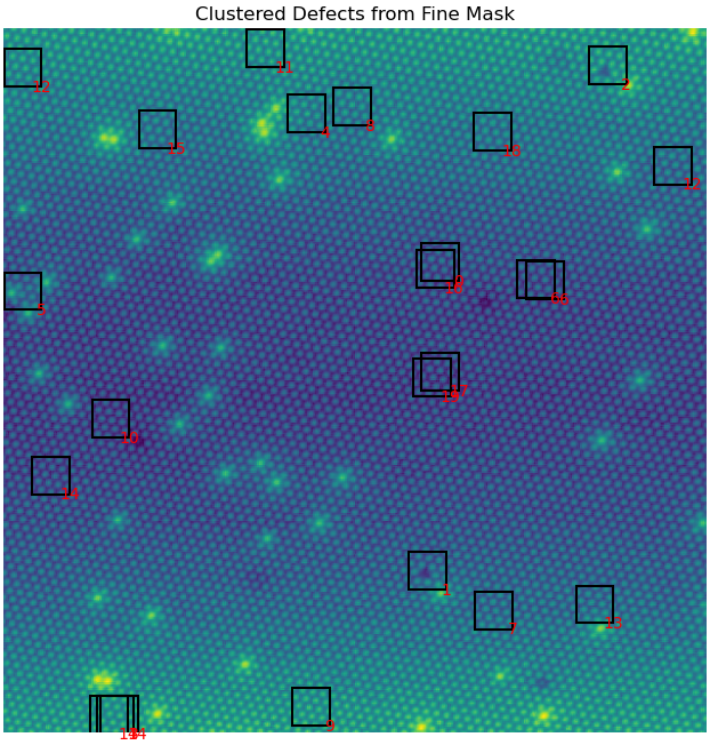


Figure 3.11: Example 1: The system correctly ignores uniform background but occasionally misclassifies low-contrast lattice noise as isolated defects. Sparse bounding boxes are inconsistently labelled, suggesting that certain morphological patterns remain underrepresented in the latent space.

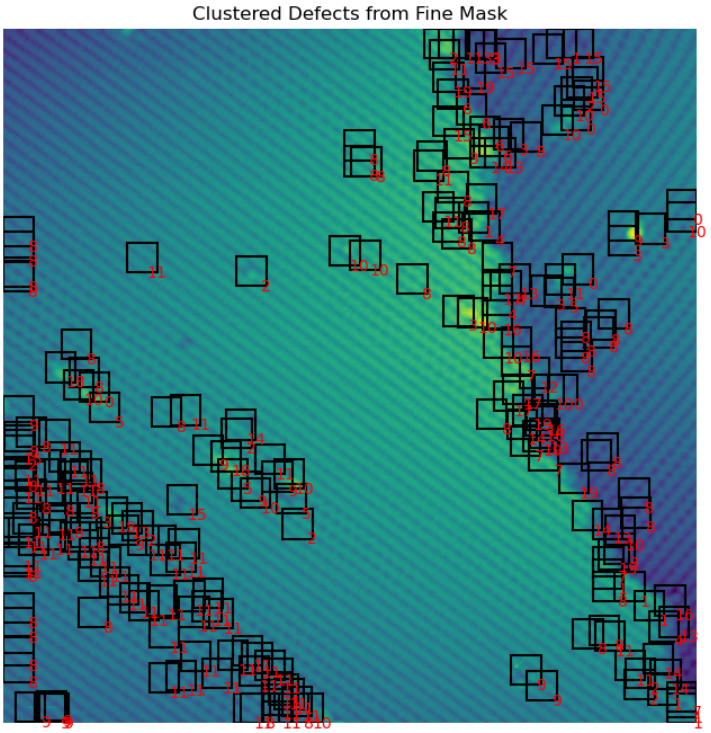


Figure 3.12: Example 2: A large atomic step edge runs diagonally through the image. The model incorrectly flags extended portions of this gradient as defective, resulting in hundreds of overlapping bounding boxes with fragmented and incoherent cluster labels.

4. Conclusions and Future Outlook

This project introduced a two-stage unsupervised learning framework for the segmentation and classification of atomic-scale features in scanning tunnelling microscopy (STM) images. By combining convolutional autoencoders with clustering algorithms, the system automatically extracted meaningful patterns from unlabelled STM data.

The first autoencoder reduced high-resolution STM patches to a 32-dimensional latent space, enabling coarse segmentation using K-means. A second, more compact autoencoder then focused on defect-rich regions, producing an 8-dimensional latent space for fine-grained clustering. Among several methods tested, Gaussian Mixture Models (GMMs) with full covariance and $k = 20$ components performed best, providing soft, interpretable labels and robust segmentation across diverse STM features. The final character maps captured key surface structures without relying on manual annotation.

Despite promising results, several limitations were identified:

Generalisation to unseen images: The pipeline performed well on familiar surfaces but failed to generalise to STM data with different surface terminations, unusual defects, or imaging artefacts. Reconstruction and clustering quality declined on out-of-distribution inputs, suggesting overfitting to the training domain.

Atomic edge-related artefacts: Regions near atomic step edges or image boundaries often produced unreliable reconstructions. These areas exhibited sharp contrast gradients or incomplete local context, which increased reconstruction error and led to misclassification. As a result, step edges were frequently flagged as defects, and valid features near the image edges were either mislabelled or omitted from the final characterisation.

To address these limitations, several improvements are suggested:

- **Use of pseudo-labels**: The K-means cluster labels from the first stage could be used as pseudo-labels to train a more accurate semi-supervised model. This would allow the system to learn more detailed features and improve consistency in classification without needing manually labelled data.
- **Physics-informed modelling**: Incorporating domain knowledge—such as surface symmetries, tip effects, or known defect types—could guide the learning process and improve robustness. This might take the form of symmetry-aware losses or structural constraints within the architecture.
- **Multi-scale representations**: Adopting transformer-based or multi-resolution architectures could help the model capture both fine-grained and extended patterns. This is particularly useful for step edges or large-area features not well represented by small patches.

Overall, this work demonstrates the viability of unsupervised learning for STM image segmentation and lays the foundation for future systems capable of general, interpretable atomic-scale analysis.

Bibliography

[1] M. Turner, "Bakeryscan and cyto-aiscan," Medium, 11 2021. [Online]. Available: https://towardsdatascience.com/bakeryscan-and-cyto-aiscan-52475b3cb779

[2] M. A. Elhassan, C. Zhou, A. Khan, A. Benabid, A. B. Adam, A. Mehmood, and N. Wambugu, "Real-time semantic segmentation for autonomous driving: A review of cnns, transformers, and beyond," *Journal of King Saud University - Computer and Information Sciences*, p. 102226, 11 2024.

[3] L. Dai, Z. Luo, and S. Li, "Exploring part features for unsupervised visible-infrared person re-identification," *Proceedings of the 1st ICMR Workshop on Multimedia Object Re-Identification*, vol. 20, pp. 1–5, 05 2024.

[4] G. Binnig, H. Rohrer, C. Gerber, and E. Weibel, "Surface studies by scanning tunneling microscopy," *Physical Review Letters*, vol. 49, pp. 57–61, 07 1982.

[5] K. Nakamae, "Electron microscopy in semiconductor inspection," *Measurement Science and Technology*, vol. 32, p. 052003, 03 2021.

[6] S. Wasswa, "How convolutional neural networks work." wasswa-sam.netlify.app, 02 2017. [Online]. Available: https://wasswa-sam.netlify.app/deeplearning/2017/02/26/how-convnets-work.html

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 04 2015.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, 05 2012.

[9] A. Lenail, "Nn svg," alexlenail.me. [Online]. Available: https://alexlenail.me/NN-SVG/index.html

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996. [Online]. Available: https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf

[11] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," *Advances in Knowledge Discovery and Data Mining*, vol. 7819, pp. 160–172, 2013.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] A. Jamal, A. Handayani, A. A. Septiandri, E. Ripmiatin, and Y. Effendi, "Dimensionality reduction using pca and k-means clustering for breast cancer prediction," *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, p. 192, 12 2018.

[14] X. Wang, J. Li, H. K. Ha, J. C. Dahl, J. C. Ondry, I. A. Moreno-Hernandez, T. Head-Gordon, and A. P. Alivisatos, "Autodetect-mnp: An unsupervised machine learning algorithm for automated analysis of transmission electron microscope images of metal nanoparticles," *JACS Au*, vol. 1, pp. 316–327, 02 2021.

[15] M. Ziatdinov, U. Fuchs, O. James, J. N. Randall, and S. V. Kalinin, "Robust multi-scale multi-feature deep learning for atomic and defect identification in scanning tunneling microscopy on h-si(100) 2x1 surface," 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2002.04716

[16] M. Ziatdinov, O. Dyck, A. Maksov, X. Li, X. Sang, K. Xiao, R. R. Unocic, R. K. Vasudevan, S. Jesse, and S. V. Kalinin, "Deep learning of atomically resolved scanning

transmission electron microscopy images: Chemical identification and tracking local transformations," *ACS Nano*, vol. 11, pp. 12 742–12 752, 12 2017.

[17] M. Ziatdinov, O. Dyck, X. Li, B. G. Sumpter, S. Jesse, R. K. Vasudevan, and S. V. Kalinin, "Building and exploring libraries of atomic defects in graphene: Scanning transmission electron and scanning tunneling microscopy study," *Science advances*, vol. 5, 09 2019.

[18] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, pp. 651–666, 06 2010.

[19] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, 1988.

[20] T. Kurihana, I. Foster, R. Willett, S. Jenkins, K. Koenig, R. Werman, R. B. Lourenco, C. Neo, and E. Moyer, "Cloud classification with unsupervised deep learning," arXiv.org, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.15585

[21] K. Kimoto, J. Kikkawa, K. Harano, O. Cretu, Y. Shibazaki, and F. Uesugi, "Unsupervised machine learning combined with 4d scanning transmission electron microscopy for bimodal nanostructural analysis," *Scientific Reports*, vol. 14, p. 2901, 02 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-53289-5

[22] J. Chen and Y. Bao, "Effect of dataset representation bias on generalizability of machine learning models in predicting flexural properties of ultra-high-performance concrete (uhpc) beams," *Engineering Structures*, vol. 326, pp. 119 508–119 508, 12 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141029624020704

[23] R. Bellman, *Adaptive control processes: a guided tour*. Princeton University Press, 1972. [Online]. Available: https://www.jstor.org/stable/j.ctt183ph6v

[24] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:5855042

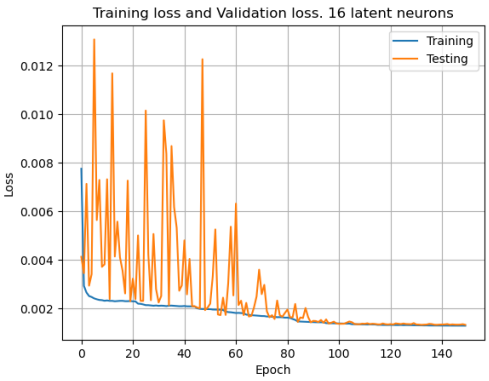
[25] E. Schubert, "Stop using the elbow criterion for k-means and how to choose the number of clusters instead," vol. 25, pp. 36–42, 06 2023.

A. Autoencoder Architecture and Parameters

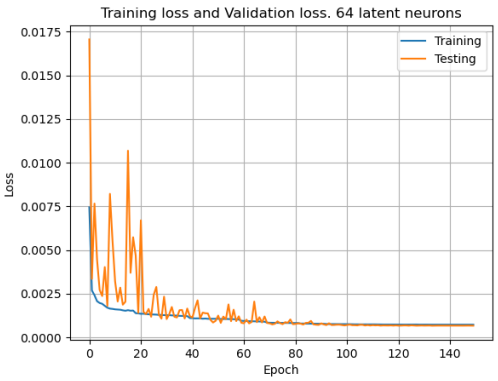
A.1. General Autoencoder Architecture

Layer (Type)	Output Shape	Parameters
Input (InputLayer)	(None, 32, 32, 1)	0
Conv2D (128 filters)	(None, 32, 32, 128)	1,280
BatchNorm	(None, 32, 32, 128)	512
ReLU Activation	(None, 32, 32, 128)	0
MaxPooling2D	(None, 16, 16, 128)	0
Conv2D (256 filters)	(None, 16, 16, 256)	295,168
BatchNorm	(None, 16, 16, 256)	1,024
ReLU Activation	(None, 16, 16, 256)	0
MaxPooling2D	(None, 8, 8, 256)	0
Conv2D (512 filters)	(None, 8, 8, 512)	1,180,160
MaxPooling2D	(None, 4, 4, 512)	0
Dropout	(None, 4, 4, 512)	0
Flatten	(None, 8192)	0
Dense (Latent Layer: 32)	(None, 32)	262,176
Dense (Upsampling)	(None, 8192)	270,336
Reshape	(None, 4, 4, 512)	0
UpSampling2D	(None, 8, 8, 512)	0
Conv2D (256 filters)	(None, 8, 8, 256)	1,179,904
BatchNorm	(None, 8, 8, 256)	1,024
ReLU Activation	(None, 8, 8, 256)	0
Dropout	(None, 8, 8, 256)	0
UpSampling2D	(None, 16, 16, 256)	0
Conv2D (128 filters)	(None, 16, 16, 128)	295,040
BatchNorm	(None, 16, 16, 128)	512
ReLU Activation	(None, 16, 16, 128)	0
UpSampling2D	(None, 32, 32, 128)	0
Conv2D (1 filter, sigmoid)	(None, 32, 32, 1)	1,153

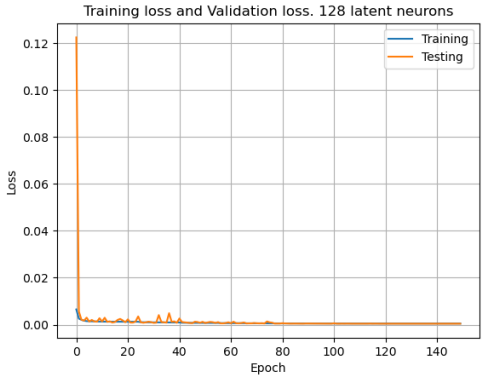
Table 1: Architecture of the convolutional autoencoder used for STM image analysis (32-neuron latent space).



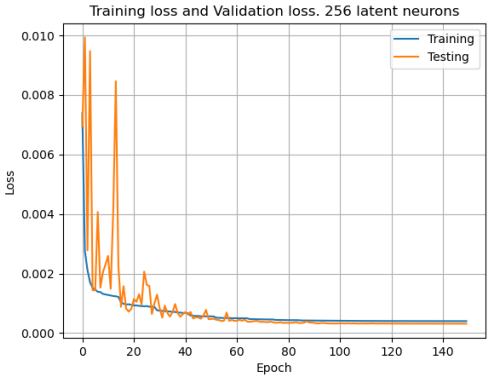
(a) Latent Dim 16



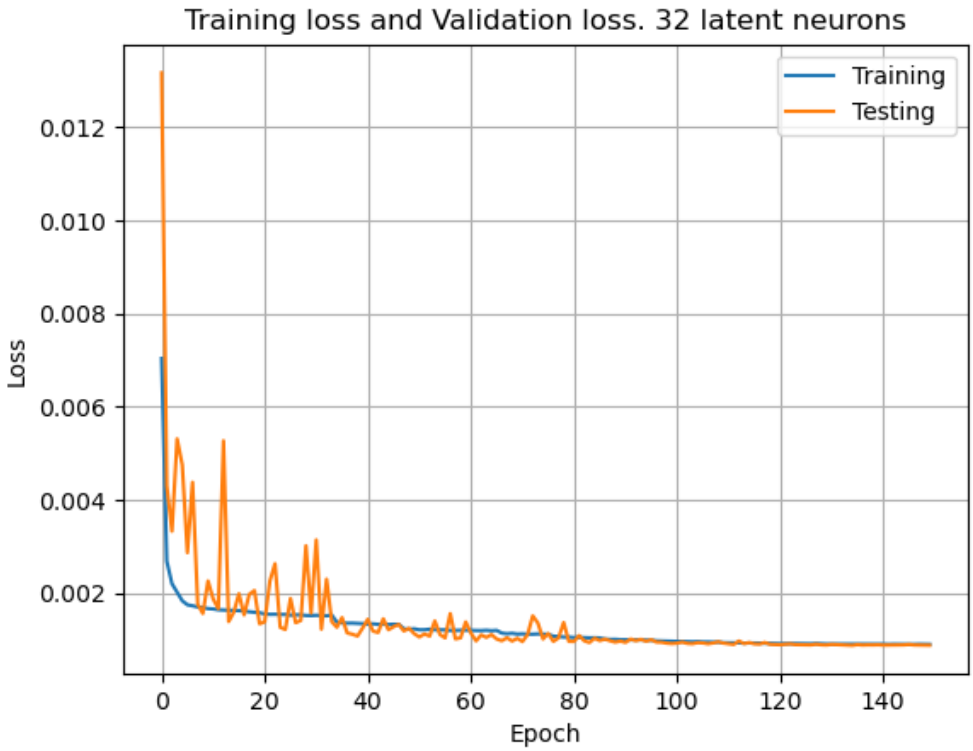
(b) Latent Dim 64



(c) Latent Dim 128



(d) Latent Dim 256



(e) Latent Dim 32 (Selected Model)

Figure A.1: Training and validation loss curves for autoencoders with varying latent dimensions. Models with higher capacity generally converge to lower final loss, though improvements beyond 32 neurons are marginal.

A.2. Defect Autoencoder Architecture

Layer (Type)	Output Shape	Parameters
Input (InputLayer)	(None, 32, 32, 1)	0
Conv2D (64 filters)	(None, 32, 32, 64)	1,664
BatchNorm	(None, 32, 32, 64)	256
ReLU Activation	(None, 32, 32, 64)	0
MaxPooling2D	(None, 16, 16, 64)	0
Conv2D (128 filters)	(None, 16, 16, 128)	73,856
BatchNorm	(None, 16, 16, 128)	512
ReLU Activation	(None, 16, 16, 128)	0
MaxPooling2D	(None, 8, 8, 128)	0
Conv2D (512 filters)	(None, 8, 8, 512)	590,336
BatchNorm	(None, 8, 8, 512)	2,048
ReLU Activation	(None, 8, 8, 512)	0
MaxPooling2D	(None, 4, 4, 512)	0
Dropout	(None, 4, 4, 512)	0
Flatten	(None, 8192)	0
Dense (256)	(None, 256)	2,097,408
Dense (Latent Layer: 8)	(None, 8)	2,056
Dense (256)	(None, 256)	2,304
Dense (8192)	(None, 8192)	2,105,344
Reshape	(None, 4, 4, 512)	0
UpSampling2D	(None, 8, 8, 512)	0
Conv2D (128 filters)	(None, 8, 8, 128)	589,952
BatchNorm	(None, 8, 8, 128)	512
ReLU Activation	(None, 8, 8, 128)	0
Dropout	(None, 8, 8, 128)	0
UpSampling2D	(None, 16, 16, 128)	0
Conv2D (64 filters)	(None, 16, 16, 64)	204,864
BatchNorm	(None, 16, 16, 64)	256
ReLU Activation	(None, 16, 16, 64)	0
UpSampling2D	(None, 32, 32, 64)	0
Conv2D (1 filter, sigmoid)	(None, 32, 32, 1)	577

Table 2: Architecture of the defect-specific convolutional autoencoder with 8-neuron latent space.

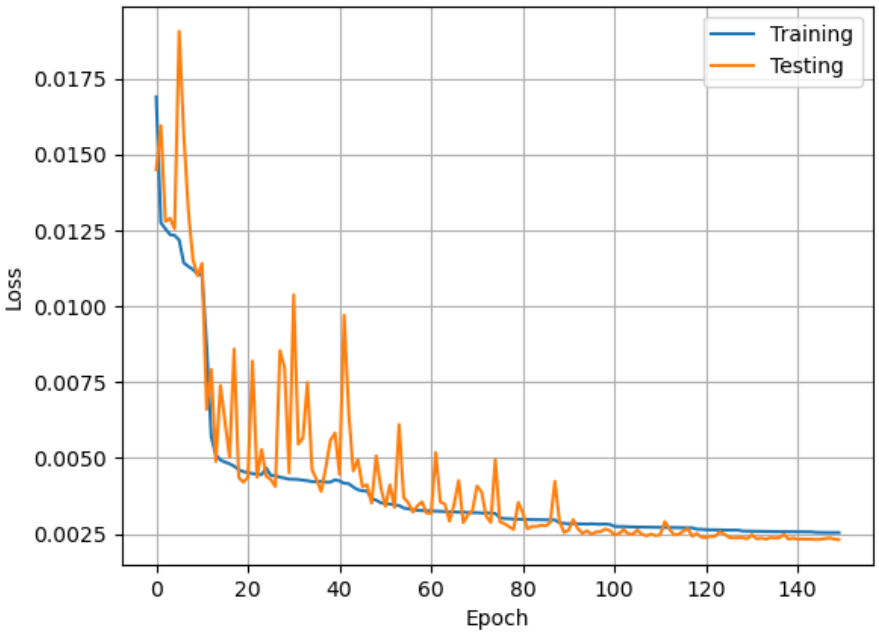
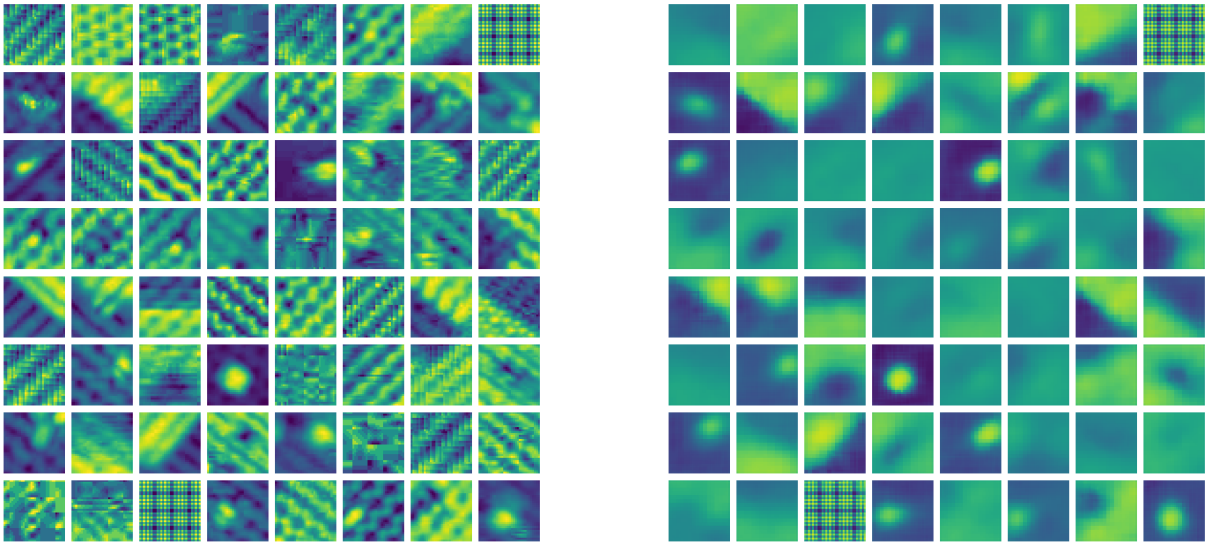


Figure A.2: Training and validation loss curves for the defect-specific autoencoder with an 8-dimensional latent space. The model was trained exclusively on patches extracted from high-error regions of STM images, allowing it to focus on rare and structurally deviant features. Convergence is reached within 30 epochs, with validation loss stabilising below 10^{-3}, indicating effective compression of atomic defect patterns without overfitting.



(a) Original patches

(b) Reconstructed patches

Figure A.3: Qualitative assessment of the 8-neuron defect autoencoder's performance. **Left**: Original STM patches extracted from high-error regions identified during the reconstruction phase. **Right**: Corresponding reconstructions produced by the compact autoencoder. Despite the aggressive compression into an 8-dimensional latent space, key atomic-scale features are preserved across a range of defect types, indicating the model's ability to capture meaningful morphology while suppressing noise.

B. Additional Figures

B.1. General Clustering Histograms

B.1.1. K-Means

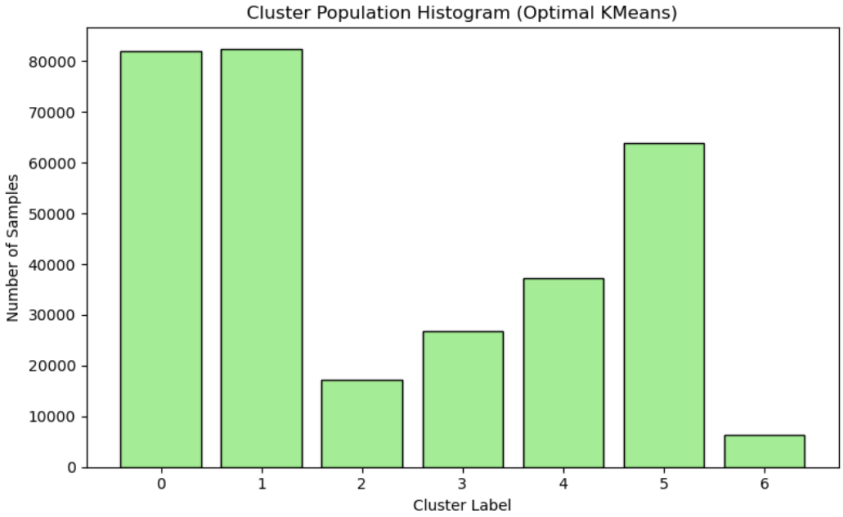


Figure B.4: Histogram displaying the frequency distribution of cluster assignments by K-means. The plot reveals the prevalence and grouping tendencies of different structural patterns in the dataset, with some clusters being highly populated while others are sparse. This distribution reflects the model's bias towards dominant surface features and may indicate under-representation of rarer defect types in the latent space.

B.1.2. DBSCAN

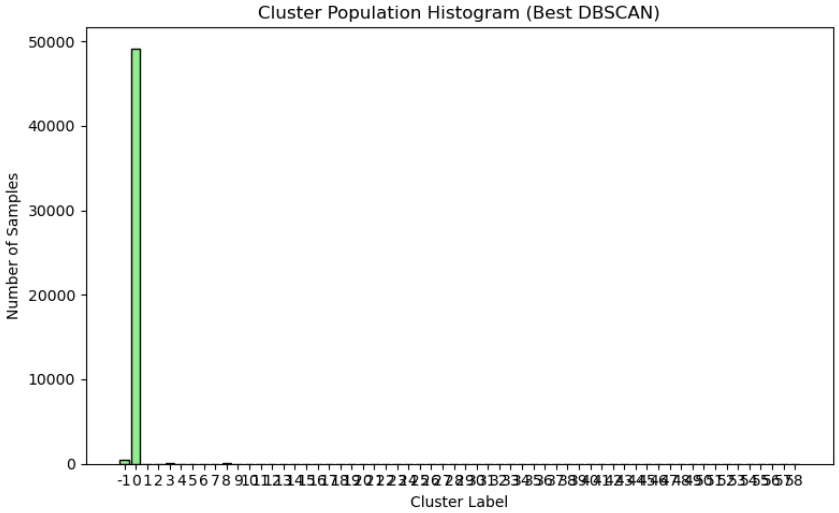


Figure B.5: Distribution of DBSCAN clustering outcomes for the general STM dataset. The vast majority of samples are either assigned to noise or absorbed into a small number of dominant clusters, with little differentiation elsewhere. This unbalanced clustering result highlights DBSCAN's high sensitivity to its hyperparameters and its limited ability to segment uniformly distributed atomic features in high-dimensional latent space.

B.2. Defect Clustering Histograms

B.2.1. DBSCAN

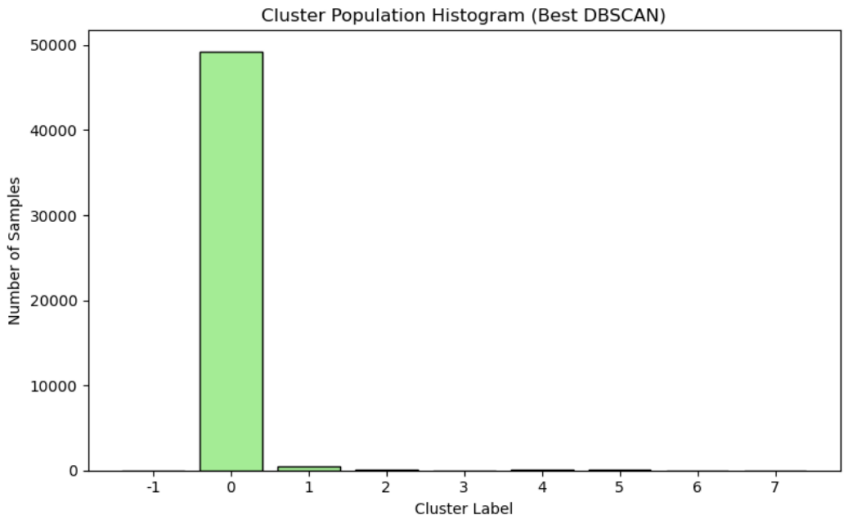


Figure B.6: DBSCAN clustering histogram for defect-specific patches, showing a high level of noise and few significant clusters, indicating the algorithm's challenges with fine-grained defect segmentation.

B.2.2. HDBSCAN

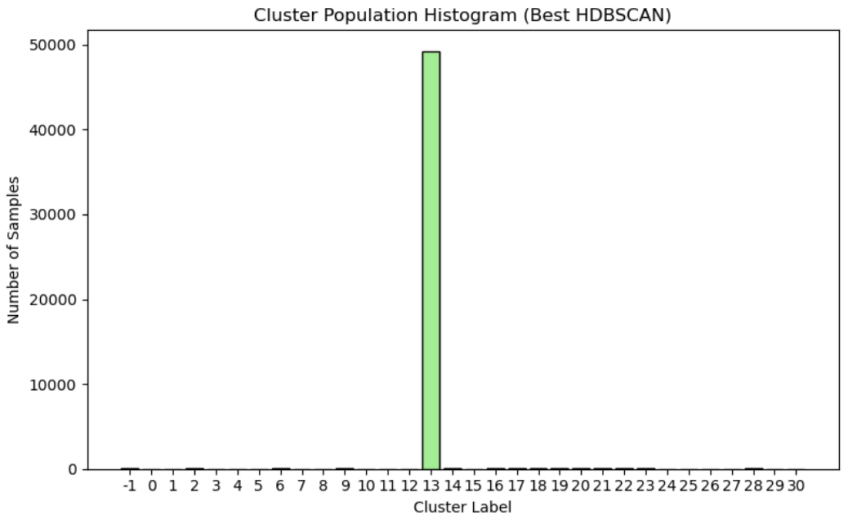


Figure B.7: Histogram of HDBSCAN results for defect embeddings, displaying better cluster stability and reduced noise compared to standard DBSCAN, suitable for the complex morphology of defects.

B.2.3. GMM

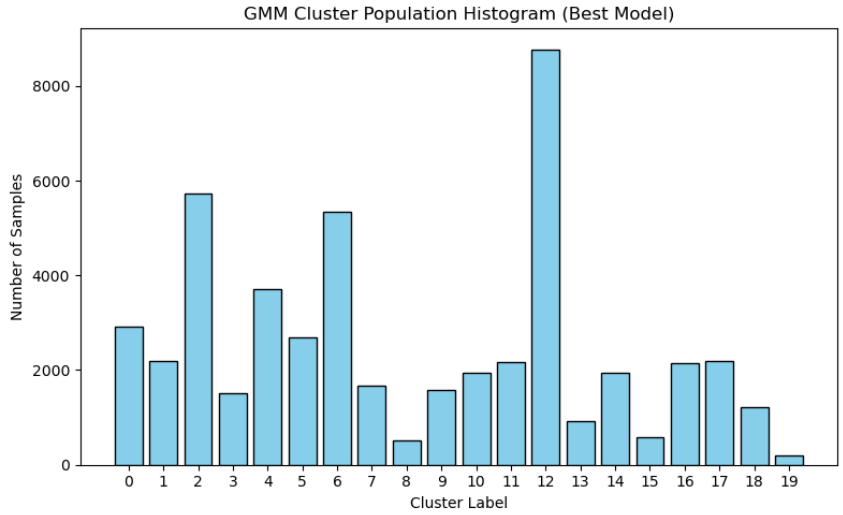


Figure B.8: GMM clustering performance on defect patches, with histogram bars representing the number of patches per cluster, indicating a well-distributed clustering across defect types.

B.3. Failure Cases Examples

B.3.1. Case Study 1

Figure B.9 shows the complete output of the pipeline for a low-defect STM image. While most of the image is correctly ignored, a few isolated patches with subtle fluctuations are misclassified (see fig. B.11). These detections are reflected in both the character map and the reconstruction error mask. Extracted patches and final groupings (fig. B.10, B.12) show inconsistent labels, underscoring challenges with under-represented defect morphologies.

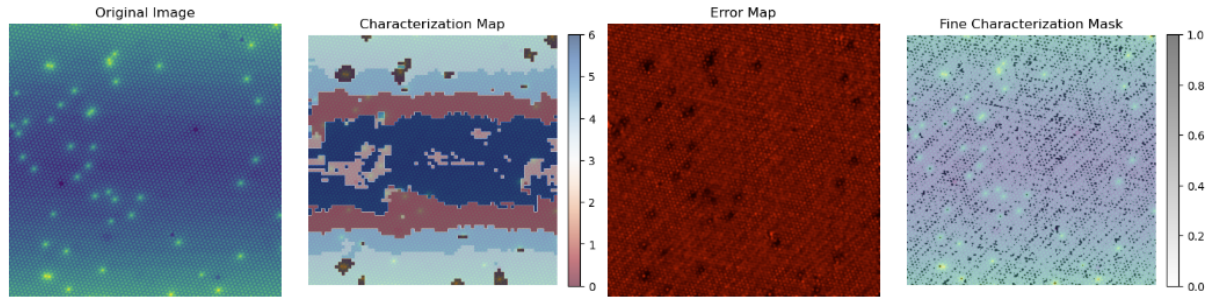


Figure B.9: Overview of the full STM image from Case Study 1. This includes the character map, reconstruction error map, fine mask, and overlay of detected bounding boxes, demonstrating the complete pipeline output on a representative test image.

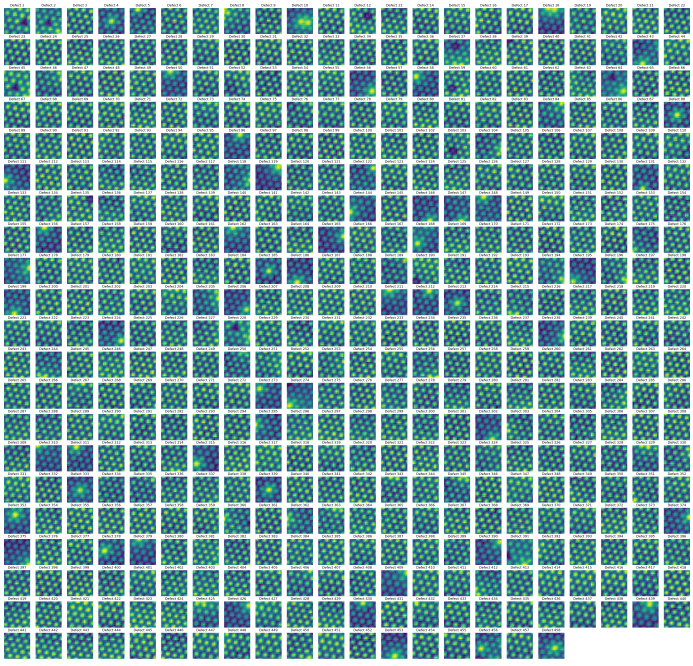


Figure B.10: Extracted patch examples from Case Study 1. Patches were selected based on the reconstruction mask and are shown with their assigned cluster labels following K-Means classification.

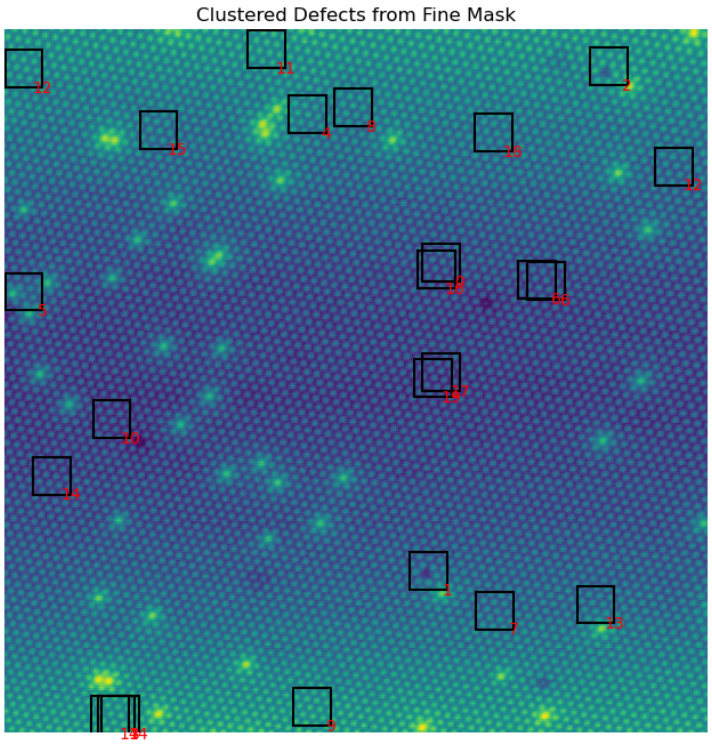


Figure B.11: Failure example: misclassification in a low-defect image. Isolated bright lattice fluctuations were flagged as defects, resulting in scattered and inconsistent cluster assignments.

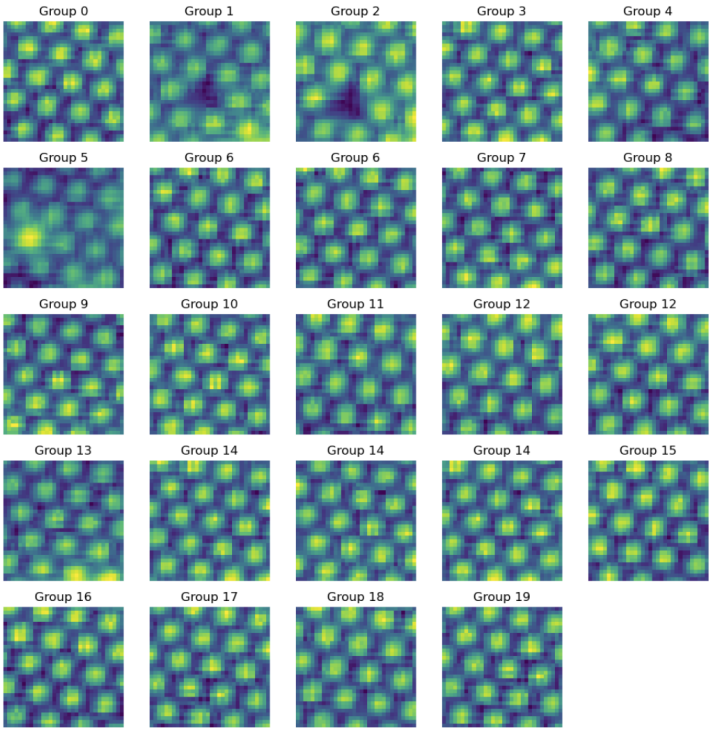


Figure B.12: Final grouped visualisation of clustered defects in Case Study 1. Distinct colours indicate different defect types as predicted by the GMM, enabling coarse-grained structural characterisation across the image.

B.3.2. Case Study 2

This image contains prominent atomic step edges. As shown in fig. B.13, the strong intensity gradients near the step dominate both the reconstruction error and defect mask. Clustering results (Appendix B.14, B.16) reveal high overlap between bounding boxes and poor cluster separation, confirming the pipeline's sensitivity to extended contrast changes.

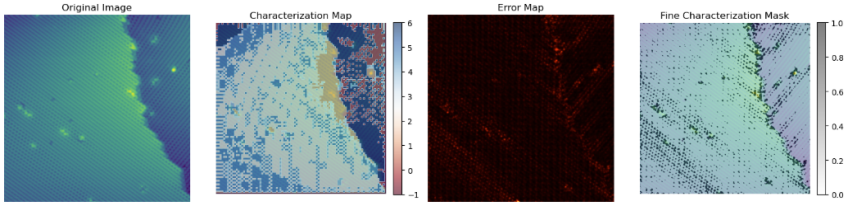


Figure B.13: Overview of the full STM image from Case Study 2. This includes the character map, reconstruction error map, binary defect mask, and overlay of detected bounding boxes, demonstrating the system's response to a structurally heterogeneous image.

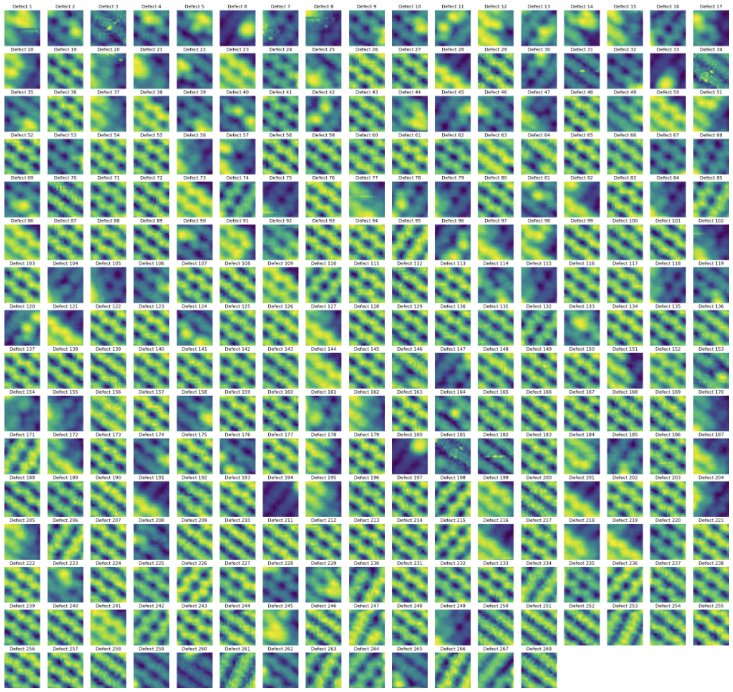


Figure B.14: Clustered patch samples from Case Study 2. Each 32×32 patch is shown with its corresponding GMM-assigned label, illustrating structural diversity among localised features.

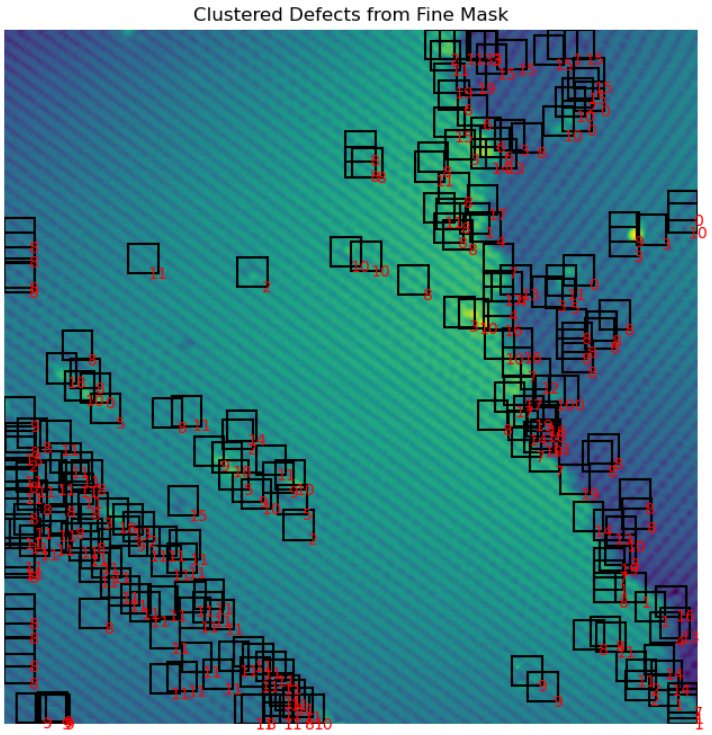


Figure B.15: Failure example: misclassification near atomic step edges. The model over-segments step edge regions, producing dense and ambiguous cluster labels due to overlapping structural patterns.

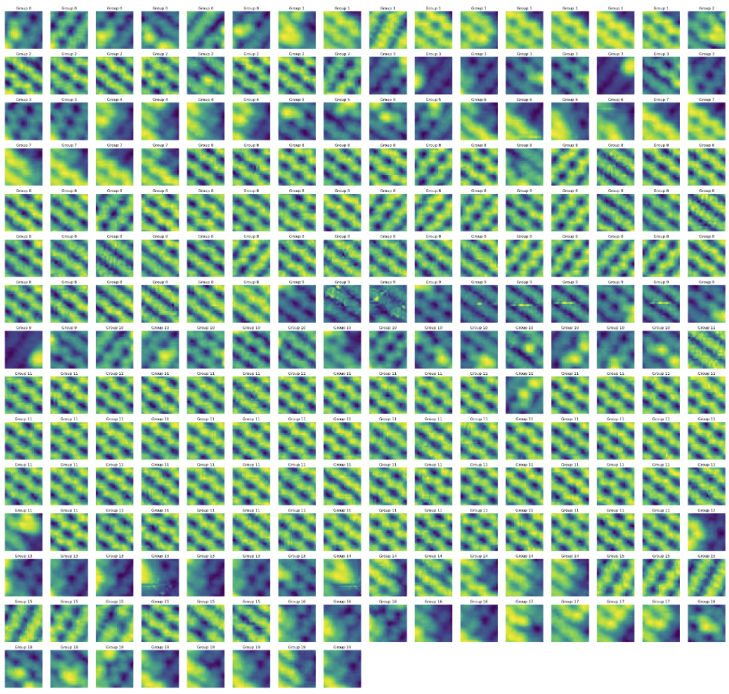


Figure B.16: Final character map of clustered defect patches in Case Study 2. Each coloured box corresponds to a GMM-assigned cluster ID, allowing coarse identification of spatial groupings and structural similarities across the image.

C. Code Availability

The full implementation of the pipeline, including data preprocessing, autoencoder training, clustering, and visualisation scripts, is available at:

`https://github.com/maxtrouble27`

The repository includes:

- Preprocessing and patch extraction scripts
- Autoencoder training code (AE1 and AE2)
- Clustering and GMM evaluation notebooks
- Visualisation utilities for generating reconstruction and character maps