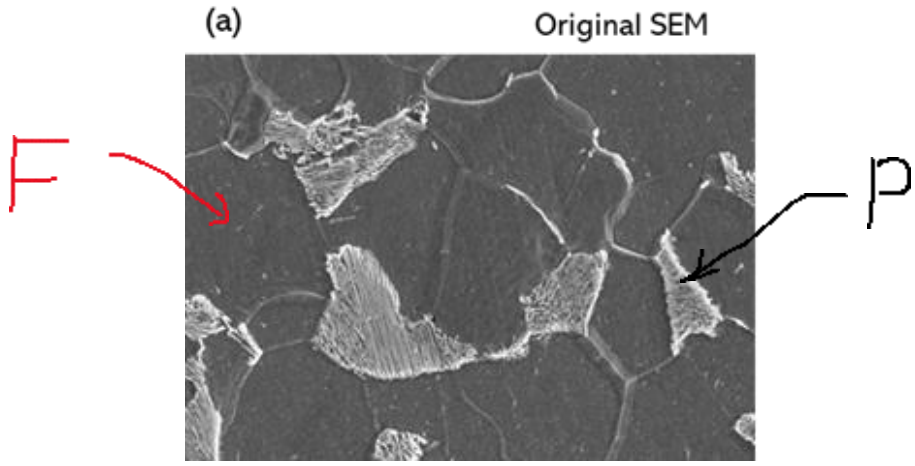
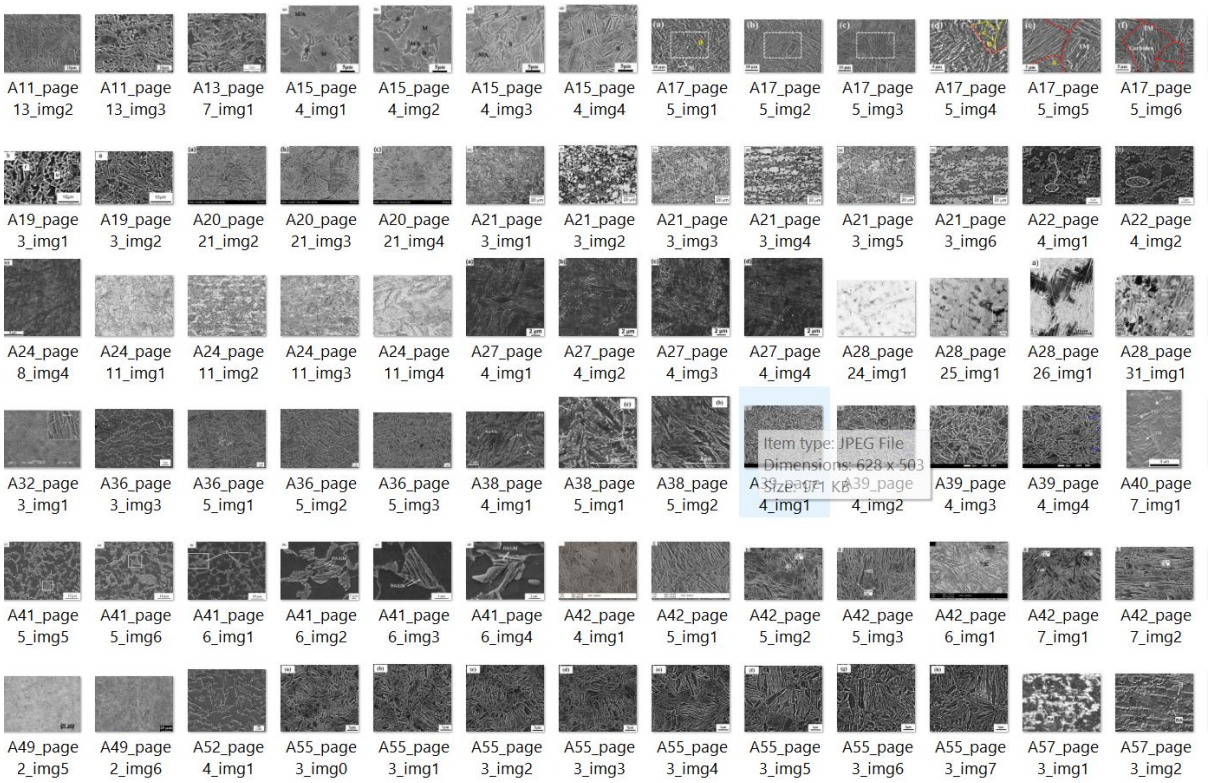
Milestone 2

2.1 Collection of suitable datasets.

341 Microstructure images have been collected after reviewing ~160 reported research articles. A sample snapshot of the data folder is presented here along with a sample microstructure. The completely collected images cannot be shared publicly as they are copyrighted with publisher.





2.2 Prepare and preprocess data for training and validation.

To confirm the dependability of the image dataset used for deep learning analysis, data preparation is an essential stage. Preprocessing in this study included histogram equalization and the removal of less-than-ideal pictures, including blurry and optical microscopic images. Histogram equalization was used to minimize the impact of fluctuating light on the segmentation and classification outcomes by standardizing the fundamental intensity features across all images. To process a grayscale micrograph, let's look at the variable I, which represents the intensities. With I = 0 denoting black and I = L-1 (i.e., 255) denoting white, an image's I should fall between [0, L-1]. If a histogram h(I) represents an image's intensity distribution, then the cumulative distribution function for each gray value (x) can be obtained as:

$$cdf(x) = \sum_{I=1}^{x} \quad h(I) \tag{1}$$

Then, the general histogram equalization $h(v)$ formula is

$$h(v) = round\left(\frac{cdf(x) - cdf_{min}}{(M \times N) - 1} \times (L - 1)\right) \tag{2}$$

where cdf_{min} the minimum non-zero value of the cumulative distribution function. M, N are the length, width of the gray-scale image. L is the number of gray levels used (i.e., 256).

Furthermore, microstructure images that were taken using optical microscopy or had a lot of blur and didn't fit the standards for clarity and resolution were routinely removed from the dataset. To prevent data leaking, the SEM dataset was split into three sections: training (70%), testing (20%), and validation (10%). The SEM-image level was used for all splits, 277 distinct micrographs were used to create the patches, and each patch is identified by its source image ID. To ensure that no patches from the same SEM micrograph appeared in more than one set, these image IDs—rather than patches—were divided into training, validation, and test sets. The testing dataset was kept apart for the sole purpose of assessing the model. To guarantee reproducibility, data augmentation was used upon splitting and the split indices were made public. The accuracy and robustness of the study's conclusions are increased by this exact preprocessing method, which guarantees that the residual images are of excellent quality and appropriate for further analytical processes.

2.3 Creation of the patches

2.3.1 Label count

A function named `label_count` is designed to increment a counter for a specific label passed to it and return the new count. However, the provided function snippet is used for global **variables**. It declares that it will use **three global variables**: `ferr_c`, `parlite_c`, and `morten_c` It takes one argument, `label_now`. It uses a series of `if` statements to check the value of `label_now` (e.g., `'ferr'`, `'parlite'`, etc.). If a match is found, it **increments the corresponding global counter** (e.g., `ferr_c += 1`). It then **returns the new value** of that specific counter.

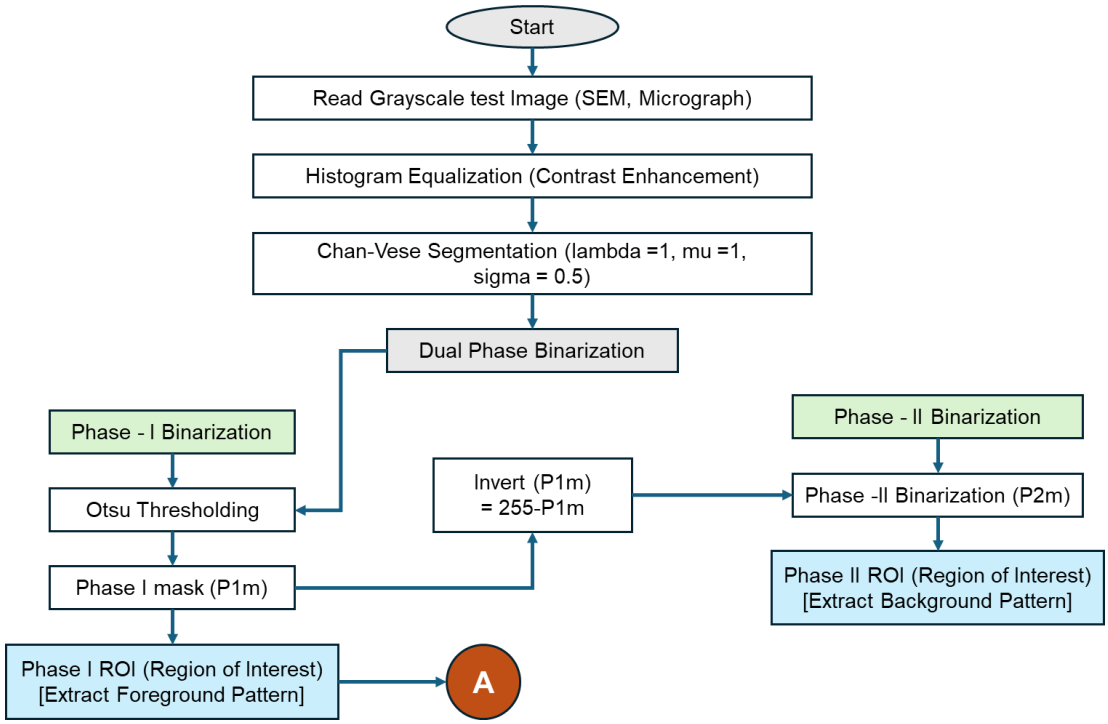
2.3.2 Largest rectangle bounding box

A Python function named `largest_rect_boundingbox` that uses the OpenCV library was used to process two input images: a binary mask (`phase_type`) and an original image (`img`). Its primary purpose is to identify the largest object (contour) in the binary mask, calculate its bounding box, enforce a minimum size for that box, and then crop the corresponding region from both input images.

2.3.3 Segmentation using Chan-Vese

The `segment(img)` function is designed to perform a complex image segmentation and enhancement workflow on an input image, likely a pre-processed (thresholded and dilated) binary or grayscale image. It heavily utilizes functions from the OpenCV and Scikit-image (`skimage` / `exposure`, `filters`, `segmentation`, `morphology`) libraries. The main goal of the function is to take an image, enhance its contrast, apply an advanced active contour segmentation method (Chan-Vese), and then use the result to generate two main binary phase masks (`phase1` and `phase2`). The first step is to load the input image (`img`) in grayscale format and immediately save it to a temporary location on the disk using OpenCV. The second step is to equalize the histogram to improve contrast, and the result is scaled back to a 0-255 range. After that, in the third primary segmentation step, which uses the Chan-Vese active contour model to find the object boundary without relying on image gradients. `Chan_Vese` is applied to the grayscale `image`. This function evolves a boundary (level set) to separate

regions with different average intensity. The key outputs are the final level set and intermediate values: the variable v is a tuple, where v[1] is the final level set (the signed distance map) and v[2] is the final binary segmentation mask.

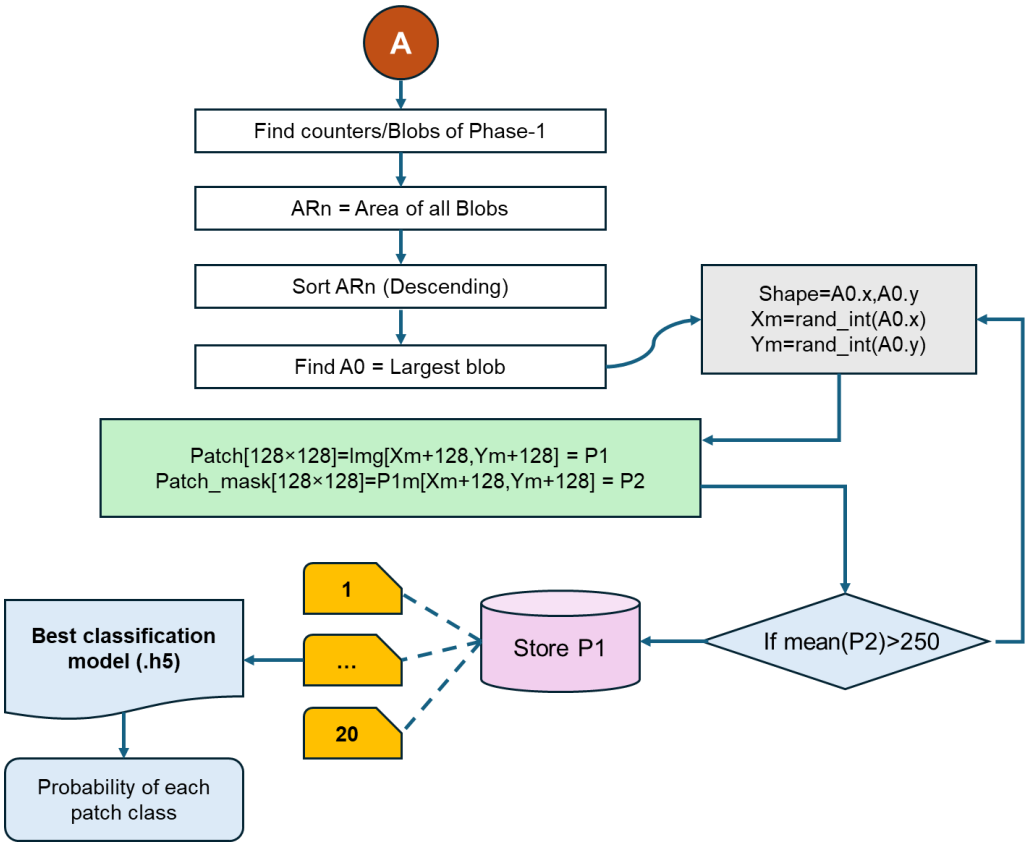


2.3.4 Refining the Segmentation Mask (Phase 1 Generation)

The segmentation result (v[1]) is further processed to create the final binary phase mask (phase1). The Initial Mask was extracted: b = v[1] from the final level set. Sign Correction was done by the calculated mean of the level set . If the mean is positive (meaning the interior of the object might be incorrectly signed), the sign is flipped: b = - b. Otsu's thresholding method is applied to the modified level set (b) to automatically find an optimal threshold, which is then used to convert the image to a binary mask. A closing operation (dilation followed by erosion) is applied with a 4 * 4 disk structuring element to smooth the boundary and fill small holes in the binary mask. The resulting refined mask is assigned to phase1. The inverse of this mask is assigned to phase2. A safety check is performed for the extreme case where the segmentation resulted in an almost entirely "on" mask. Full Image Check: If the mean of phase1 is greater than 0.999 (meaning it's essentially a solid white image), both phase1 and phase2 are manually reset to be solid white arrays of ones (of type uint8). This overrides the segmentation result if it fails dramatically.

2.3.5 Final Patch creation

This describes a complete data generation pipeline for a machine learning model. It would likely be used by Convolutional Neural Network (CNN) for classifying microstructure image patches of materials. It iterates through a list of image files, processes each one to identify and segment two distinct phases, and then randomly extracts and saves (128 * 128) pixel patches from these phases as training data, labeling them based on a predefined rule set derived from the filename. Two identical loops, one for each phase, to create a target number of training patches three is the default target. Patch creation is skipped if the phase mask's mean intensity is very low. The code randomly selects 128 * 128 patches from the current cropped phase mask phase1 or phase2 as long as the image is large enough. A patch is only accepted if its mean intensity is above a dynamic start_thresh (initially 250). This ensures the patch is predominantly composed of the phase of interest (a high-quality, non-mixed patch).



2.4 Example of patch snippets