ACCORDs - Rapport (R. Bugiel)

(15.04-15.10.2025)

1. Introduction

The ACCORDs project addresses the challenge of automated analysis of digital images of graphene-based materials. Manual categorization of such images is time-consuming, subjective, and inconsistent, especially when multiple overlapping classes must be annotated on the pixel level. The ultimate goal of this work is to enable image analysis in the context of biological risk assessment, where structural and morphological information extracted from microscopy images contributes to evaluating the potential impact of graphene-based materials on biological systems. The project is primarily focused on developing a neural network–based tool to support biological risk assessment rather than pursuing fully automated image analysis. Given the complexity of graphene images and the need for the highest level of accuracy in risk evaluation, a human-in-the-loop approach is considered a central objective of the AI-based workflow.

To achieve this, a multi-stage pipeline was designed and tested, covering:

- dataset preparation and preprocessing,
- image normalization and probability map generation,
- segmentation through both classical and deep learning methods,
- object- and image-level analysis,
- and, ultimately, linking image-derived features to their biological meaning.

My contribution included:

- developing guiding rules for image categorization,
- reviewing and integrating labeling tools (Label Studio, Ilastik, SAM2),
- creating a structured dataset preparation pipeline with normalization, probability maps, and multi-label mask generation,
- implementing PDF reporting and automated quality checks,
- training deep learning models (ResNet for classification, U-Net for segmentation),
- exploring semi-automatic image analysis based on masks

This report summarizes the methodology, tools, and results achieved, and outlines recommendations for the next steps in ACCORDs.

2. Image Categorization

Image categorization is designed as a complementary/parallel step to semantic segmentation performed by the U-Net model. While neural network outputs enable quantification of particle number and size, higher-level image descriptors (morphology, material composition, nanostructures) are derived by combining segmentation results with manual or automated labeling rules. These descriptors are subsequently linked to biological interpretation, supporting the assessment of potential biological risks. To learn about image content (starting from TEM images in this project part), hand-categorization was done on 5 images and passed to microscopist for check and validation

The categorization relies on five guiding questions:

Particle Count

- **Part or more than one fragment** – image contains only fragments of particles, or a full particle together with additional fragments. If a complete particle and part of another are present, the image falls into this class.
- **One** – a single complete particle is visible, with all edges clearly defined.
- **Few (2–10)** – two to ten complete particles are visible. Accidental holes in the carbon support film must not be counted as particles.
- **Many (>10)** – more than ten complete particles are present.

Material Composition (chemically)

- **One** – only a single type of material (e.g., GO or G). The carbon support film or fragments of it are not counted as separate materials.
- **More than one** – at least two distinct materials are present (e.g., GO and G, or GO and impurities).
- **Obvious impurities/contaminants** – clear presence of foreign or contaminant structures.
  NA – structures are visible but cannot be clearly identified as any defined material.

Morphology

- **Thick, flat, rigid** – graphene-based structure with >10 layers; appears large, flat, blocky, with sharp edges and strong contrast.
- **Thin, flat or wavy, with few kinks** – <10 layers; semi-transparent regions with smooth or gently undulating surfaces.
- **Thin, crumpled, with many folds** – irregular, wrinkled textures resembling a crumpled film.
- **Combination** – two or more of the above morphologies co-occur in the same image.
- **Other** – structures unlike flakes (e.g., fibrous, spherical, or unusual textures).
- **Disorganized graphene** – patchy, noisy appearance lacking a clear flake shape; amorphous character.
- **NA** – unclear, noisy, or ambiguous image where morphology cannot be reliably identified.

Nanostructures

- **None** – no nanostructures are visible.

- **One** – a single nanostructure is present.
- **Few (<10)** – between two and ten nanostructures are present.
- **Many (>10)** – more than ten nanostructures are present.

A substantial proportion of TEM images are difficult to categorize unambiguously. To ensure consistent labeling during training dataset preparation, a dedicated guide (*TEM_char_guide*) has been developed. This document provides a set of guiding questions for classification and addresses common doubts in categorization. For each magnification level (2000x, 10000x, 50000x, 100000x, 50000x, 1200000x), the guide includes representative image examples and illustrates how the questions should be applied — such as determining the number of particles, identifying material composition, or describing morphology.

/This is work in progress and a lot of images should be added there together with specialist verification./

To learn about images (starting from TEM images in this project part), hand-categorization was done on 5 images as guided above and passed to microscopists for check and validation.

/Wating for Kerstin Jurkschat output on that. This part is also not suitable for an article. But is covering the work that other partners demanded and was not finished because of the lack of specialist input./

Transfer learning for image categorization

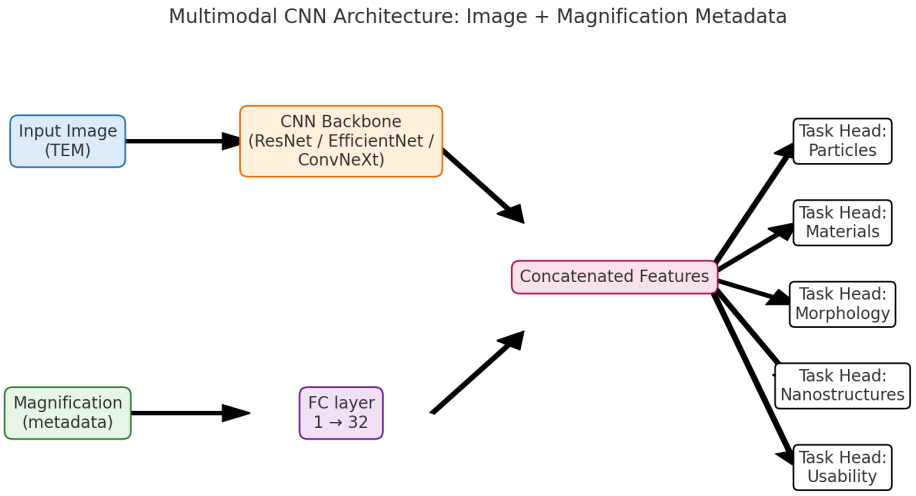

Figure 1 Model architecture

The classification task was implemented in a custom PyTorch pipeline. Transmission electron microscopy (TEM) image folders were paired with structured label sheets in `.csv` or `.xlsm` format, where each image was annotated with five descriptors corresponding to predefined categorization questions. These descriptors were encoded as integer classes and merged into a single dataset for model training. In total, 185 manually classified images were used.

A custom dataset class was developed to link image files with their encoded labels and apply standard preprocessing steps such as resizing, normalization, and augmentation. The dataset was stratified into training and validation subsets to preserve label distribution, and PyTorch DataLoaders were used to enable efficient batch processing. This configuration allowed the model to take raw TEM images as input and output predictions across all five categorical dimensions, directly comparable to manual annotations.

Stage	Description
Input	TEM image folders (.tif/.jpg/.png) with associated label sheets (.csv/.xlsm).
Preprocessing	Merge label sheets; encode textual categories into numeric classes; resize and normalize images; scale magnification values for use as metadata input (from file names)
Model	PyTorch multi-task **multimodal classifier** with CNN backbones (ResNet, EfficientNet, ConvNeXt) using transfer learning. Image features are concatenated with encoded magnification metadata
Training	Stratified train/validation split; DataLoaders with standard augmentations; optimization with cross-entropy losses.
Output	Predicted categories for each image across five dimensions (particle count, material type, morphology, nanostructures, usability).

Table 2. Overview of the image classification workflow.

To account for the strong dependence of observable features on magnification, the model was extended with a multimodal input design. In addition to convolutional backbones that extracted image features, magnification metadata was encoded through a fully connected layer (1 → 32 units) and concatenated with the visual feature vector. This joint representation was then passed to the classification heads, allowing the network to learn magnification-specific priors. For example, at 2 000× only particle counts can be reliably observed, whereas morphology and nanostructures can be distinguished only at higher magnifications.

The classification experiments employed a multi-task convolutional neural network with pretrained backbones, including ResNet18, ResNet34, EfficientNet-B0, and ConvNeXt-Tiny,

all implemented in PyTorch. Each model was initialized with ImageNet weights for transfer learning and extended with task-specific fully connected heads to predict the five categorical outputs: particle count, material composition, morphology, nanostructures, and usability.

ConvNeXt-Tiny and EfficientNet-B0 achieved the highest overall validation accuracy (~77%), outperforming the other backbones across most tasks. ResNet18, despite being the smallest architecture, produced competitive and stable results. Accuracy was highest for usability (>96%) as well as for number of materials and particle count (~90%). Morphology proved to be the most challenging category (<45%), reflecting the intrinsic ambiguity of manual labels. Nanostructure classification also remained difficult, primarily due to the limited number of training examples.

These findings demonstrate that pretrained CNN backbones transfer effectively to the categorization of TEM images, with ConvNeXt-Tiny offering the best balance between accuracy and robustness. However, performance is constrained by the quality of manual annotations. Morphological classification is inherently subjective, even for human experts, and mislabeled data can confuse the model. Involving domain specialists, expanding the dataset, and refining class definitions in “difficult” categories represent the most promising directions for improving classification accuracy in future work.

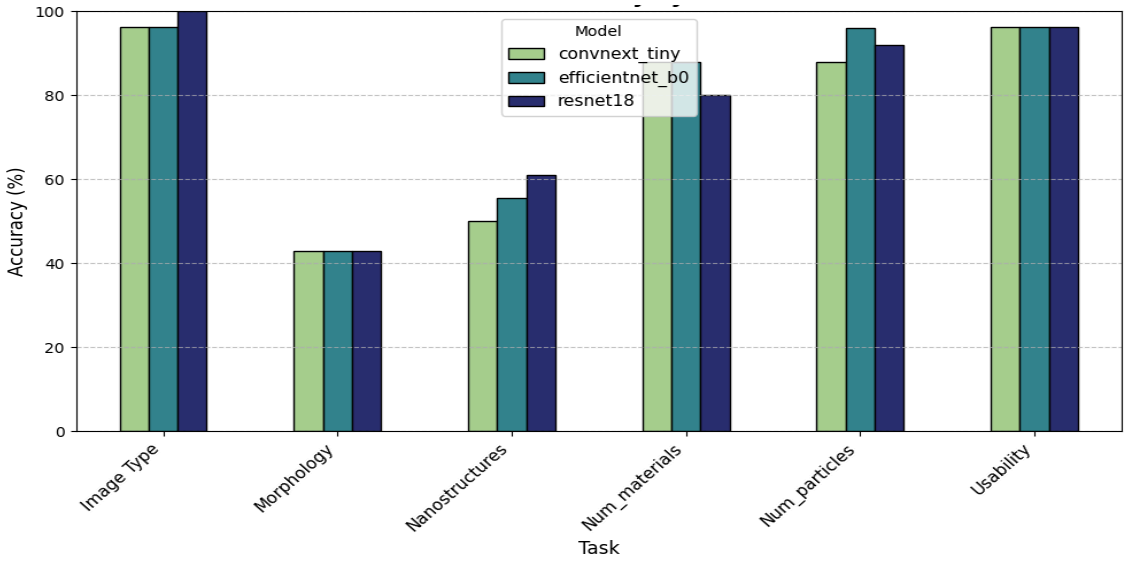

Table 4. Validation accuracy shown per class and per model.

Task	ResNet18	EfficientNet-B0	ConvNeXt-Tiny
Particle count	88.00	96.00	92.00
Number of materials	88.00	96.00	92.00
Morphology	42.86	42.86	42.86
Nanostructures	50.00	55.56	61.11
Usability	96.15	96.15	96.15

Average	**73.00**	**77.31**	**76.82**

Table 3. Validation accuracy (%) per task and backbone.

The notebook is available here:
A_Multimodal_MultiTask_Classification_of_TEM_Images_PyTorch

3. Evaluation of labeling tools

Accurate pixel-level labeling of graphene flake material (GFM) in TEM images presents several specific challenges. The GFM structure is highly heterogeneous, with irregular edges and asymmetric morphology. Each pixel must be correctly assigned to its class, yet many images at larger scales also include carbon support holes. These holes are overlapping with the GFM and may introduce artificial edges or local contrast variations. While such holes could theoretically be excluded from analysis, their overlap with flakes risks introducing artifacts that may bias the morphological interpretation. Furthermore, the number of objects per image is very large, requiring semi-automated workflows with the option for manual correction. In many cases, object boundaries are not sharply defined, making probabilistic labeling (pixel-wise likelihood of class membership) more suitable than strictly binary masks. Finally, the labeling software should remain lightweight and efficient to support practical use.

We tested several software packages with these requirements in mind:

1. **Label Studio**
   Label Studio was tested as a web-based annotation platform, installed under Windows within a Conda environment. The interface is relatively lightweight and simple, though not entirely intuitive. A limitation was the lack of direct TIFF support, requiring either a plug-in or external conversion. Pixel-wise multi-labeling is technically possible but relies on manual painting, which is slow and impractical for large datasets.

   We attempted to integrate Label Studio with the SAM (Segment Anything) model via a custom Flask backend, aiming to accelerate semantic segmentation through

automatic pre-labeling with the possibility of manual correction. The backend was successfully configured, responding correctly to prediction requests, and local storage integration was established for high-resolution SEM images. However, the SAM integration failed to return visible masks, preventing full automation In practice, annotations could still be drawn by hand, but this process was inefficient for dense TEM images. Overall, although Label Studio offers flexibility and the potential for integration with modern segmentation models, its configuration was non-trivial and performance on Windows was unsatisfactory. For these reasons, we abandoned it as the primary tool in the current workflow.

The API responded with None. Since it might be a technical issue to be solved via professionals, the Label Studio did not pass a test for “easy-to-run” software as needed.

2. **LabelMe**
   LabelMe provided a straightforward and user-friendly annotation interface and was generally well suited for polygon-based labeling tasks. A useful feature was the possibility to draw polygons assisted by AI (SAM-based), which accelerated annotation compared to purely manual drawing. However, the tool lacked integration with OMERO, limiting its direct applicability for microscopy data management. Furthermore, the software proved unstable when applied to larger TEM datasets, frequently crashing and thereby restricting its reliability for systematic labeling. While LabelMe remains attractive for small-scale experiments, its limitations in stability and interoperability make it unsuitable for high-throughput workflows in the ACCORDS project.

3. **Fiji/ImageJ and Napari**
   Both Fiji and Napari provide powerful plugin ecosystems and advanced visualization capabilities. Fiji is a long-established standard in microscopy image analysis, while Napari offers modern integration with Python workflows. However, both platforms are primarily designed for image analysis rather than efficient mask preparation, and in our tests they proved too heavy for high-throughput annotation tasks, with considerable computational overhead and slower performance relative to the project needs. Nevertheless, given their extensibility and active development communities, we recommend further exploration of Fiji and Napari for future applications in mask generation and annotation support.

4. **Ilastik**
   Ultimately selected as the preferred tool. Ilastik provides an interactive, machine-learning-based labeling workflow with probabilistic pixel classification, which addresses the ambiguity of poorly defined edges. It supports semi-automatic segmentation with manual correction, is relatively lightweight, and can efficiently process large image sets. These features made it the most suitable tool for GFM segmentation in the ACCORDS project.

Workflow	Purpose & Advantages
Pixel Classification	Probabilistic pixel labeling via Random Forest. interactive and batch-capable.
Autocontext	Improves pixel labeling through cascade-based refinement.
Object Classification	Classifies segmented objects using shape and intensity features.
Boundary-based Multicut	Optimized boundary-guided segmentation, ideal for EM-style, closed-surface objects.
Density Counting	Fast object counting in crowded images without segmentation.
Carving	Semi-automatic segmentation via interactive seeding, suitable for complex shapes.

Table 4. Selected workflows in Ilastik. Up to now only pixel classification is used.

Segment Anything Model (SAM)

The SAM2, developed by Meta AI, represents a new class of foundation models for computer vision. It was trained on over one billion segmentation masks across diverse datasets and is designed to perform *prompt-based segmentation* without domain-specific retraining. In practice, SAM can generate object masks from minimal input, such as points or bounding boxes, and in fully automatic mode it proposes multiple candidate segmentations for a given image. This general-purpose capability makes it an attractive candidate for material science applications, where manual annotation remains highly time-consuming.

A first exploratory notebook was developed to test SAM on TEM/SEM images of graphene flakes within the ACCORDS project. The implementation was based on the official Segment Anything (SAM) Python package released by Meta AI, installed directly in the Colab environment. The notebook used the *SamAutomaticMaskGenerator* class to apply SAM in automatic mode on graphene TEM/SEM images.

This initial experiment confirmed that SAM is able to detect a large number of objects in graphene images (e.g., 272 objects in a single high-resolution micrograph), demonstrating strong potential for semi-automated labeling. The example is shown on Figure X.

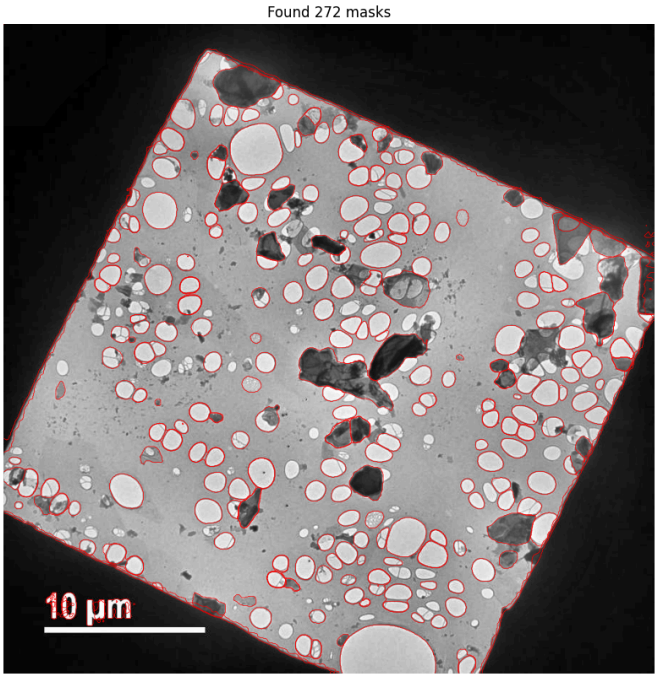

Figure X. TEM image after segmentation using SAM2. Only contours of detected objects are shown for better visibility.

At the same time, the trial highlighted several limitations that require further work:

1. **Distinguishing flakes vs. carbon support holes**
   SAM returns generic object masks without semantic labels. For the ACCORDS context, objects must be separated into several classes (in the simplest approach GFM, undefined objects and carbon support holes, but in the further extension GFM should be divided into morphologically and chemically specific graphene objects).
   Potential solutions include post-classification of masks using shape, intensity, and texture features, followed by lightweight machine learning methods such as k-Nearest Neighbors or Random Forest.

2. **Analysis of segmented objects**
   Once reliable classification is achieved, masks can be quantified in terms of area, perimeter, circularity, and texture. This will allow systematic comparison of morphological descriptors across magnifications and samples

3. **Technical improvements to the notebook**
   The current prototype was developed on Google Colab (free tier), which proved unstable for high-resolution TEM images (frequent memory crashes). Migration to a more robust environment, such as Workbench AI or cloud-based GPU instances, is required for systematic use.

   Several enhancements should be added, including preprocessing (contrast normalization, denoising), tiling of large TIFFs, post-processing of masks (morphological cleaning), and quantitative evaluation against ground truth labels.

Such steps have been implemented for other workflows (see Section X) and simply can be adapted here.

4. **Coverage of objects**
   Not all flakes or holes were captured by SAM, particularly those with low contrast or partially overlapping structures.
   This can be addressed by adjusting SAM parameters (e.g., number of sampling points, IoU thresholds), using prompt-based segmentation for missed structures, or complementing SAM with domain-specific models fine-tuned on graphene images.

Overall, this first experiment with SAM on graphene TEM/SEM images is promising but preliminary. It demonstrates the feasibility of leveraging foundation models to accelerate annotation but also underlines the need for domain-specific adaptations.

The notebook is available here: B_Segment-Anything-SAM2-TEM-Graphene

/I did not continue this step because in time we didn't have cloud resources for that and simply free notebook crashes constantly. Currently, it could be developed in tested with more resources since results is promising/

3.Training Dataset Preparation Pipeline

The full pipeline is shown on Figure Y.

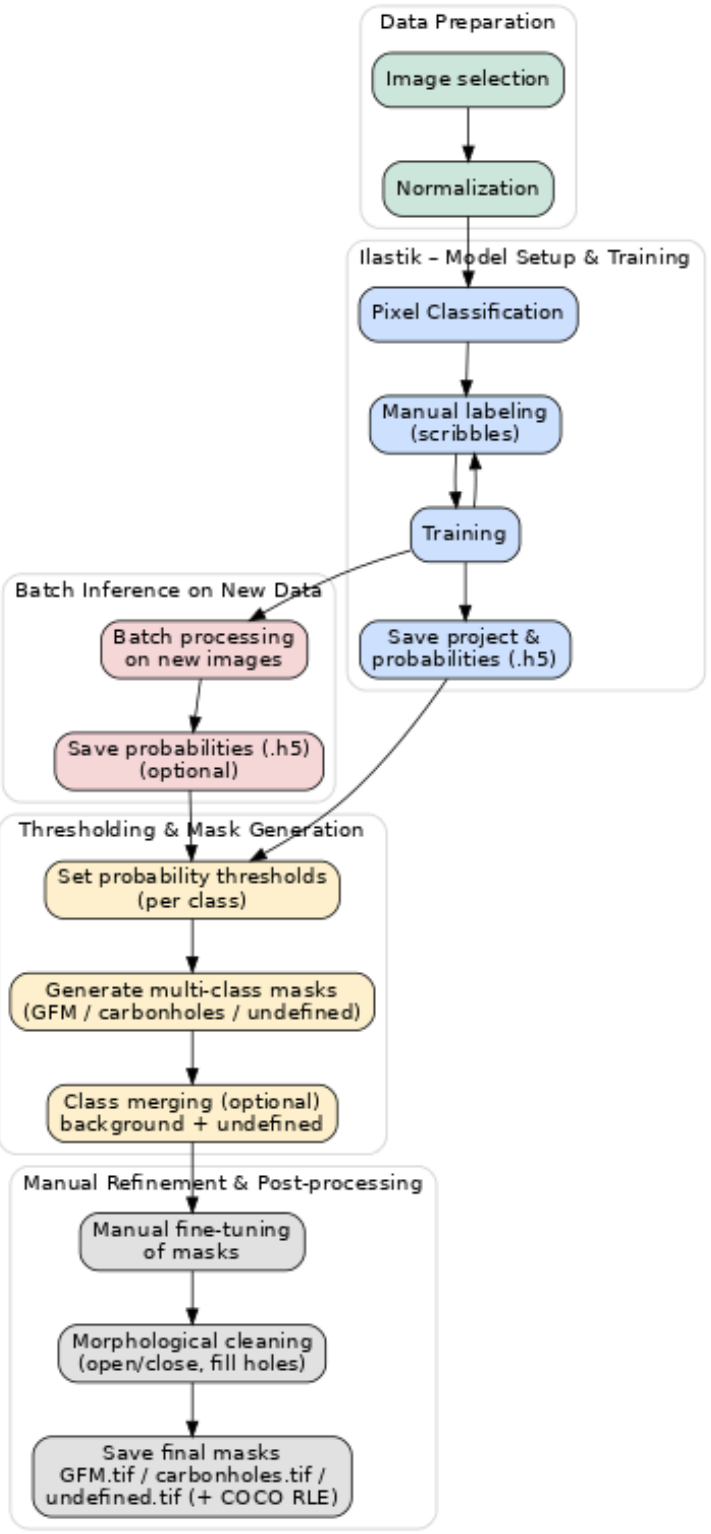

Figure X. Scheme of training dataset preparation pipeline. **/Should be tuned a bit/**

The preparation of the training dataset followed a structured pipeline designed to ensure reproducibility and consistency across all TEM images.

The first step involved image selection and normalization. Raw TEM images were collected and normalized to reduce variability in brightness and contrast introduced by acquisition conditions. This provided a standardized input for subsequent pixel classification.

The next stage was Ilastik pixel classification. Using Ilastik's interactive workflow, pixel classification models were trained on a small set of manually annotated images. The trained models produced probability maps for each class of interest: *graphene flake material (GFM)*, *carbon support holes*, and *undefined*.

After this initial Ilastik session, the pipeline continued with thresholding and primary mask generation. Probability maps were converted into binary masks by applying class-specific and image-specific thresholds. The primary masks could then be reloaded into Ilastik, where optional batch processing was applied to new, unseen images.. Depending on the quality of the results, the Ilastik and thresholding stage could be repeated to further refine the outputs.

Finally, the workflow included manual refinement and post-processing. Ambiguous areas were corrected in Ilastik, while additional post-processing steps (e.g., morphological cleaning) were implemented in a standalone Python notebook.

Normalization

Before pixel classification and mask generation, all TEM images were normalized to reduce acquisition-related variability in brightness and contrast. A dedicated Python pipeline was developed to apply and compare two normalization approaches:

1. Min–Max normalization
   This method rescales image intensities linearly such that the minimum pixel value is mapped to 0 and the maximum to 255. While simple and widely used, in practice it often introduces negligible changes in TEM data. This is because the raw images already span the full 8-bit range, or because outliers determine the global minimum and maximum, leaving most pixel intensities unaffected.

2. CLAHE (Contrast Limited Adaptive Histogram Equalization)
   In contrast to global min–max scaling, CLAHE enhances contrast locally. The image is divided into tiles, and histogram equalization is applied within each tile with a predefined clip limit to avoid noise amplification. This method improves the visibility of fine structures such as graphene flake edges, even when global intensity differences are small.

In practice, min–max normalization produced little to no improvement, while CLAHE significantly increased local contrast and was therefore selected as the default preprocessing step for subsequent analysis. The comparison of both approaches is shown in Figure Y.

The notebook is available here: C_tem_image_normalization_clahe_minmax

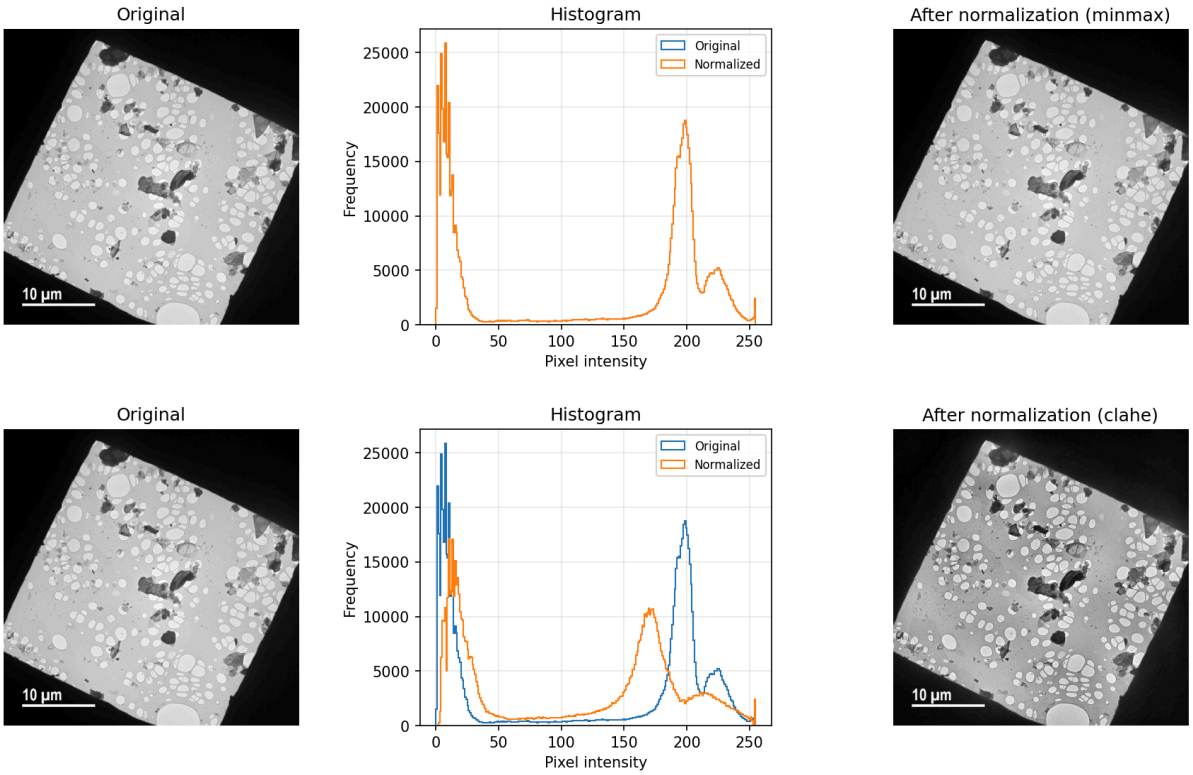

Figure X. TEM image normalization. Top: min–max scaling; bottom: CLAHE. Each panel shows the original image (left) and the normalized image (right), with corresponding histograms of pixel intensity distributions (center). For min–max scaling, the intensity distribution remains unchanged because the original image already spans the full 0–255 range. CLAHE, in contrast, substantially enhances local contrast and improves the robustness of Random Forest–based classification in Ilastik.

Ilastik-based mask preparation

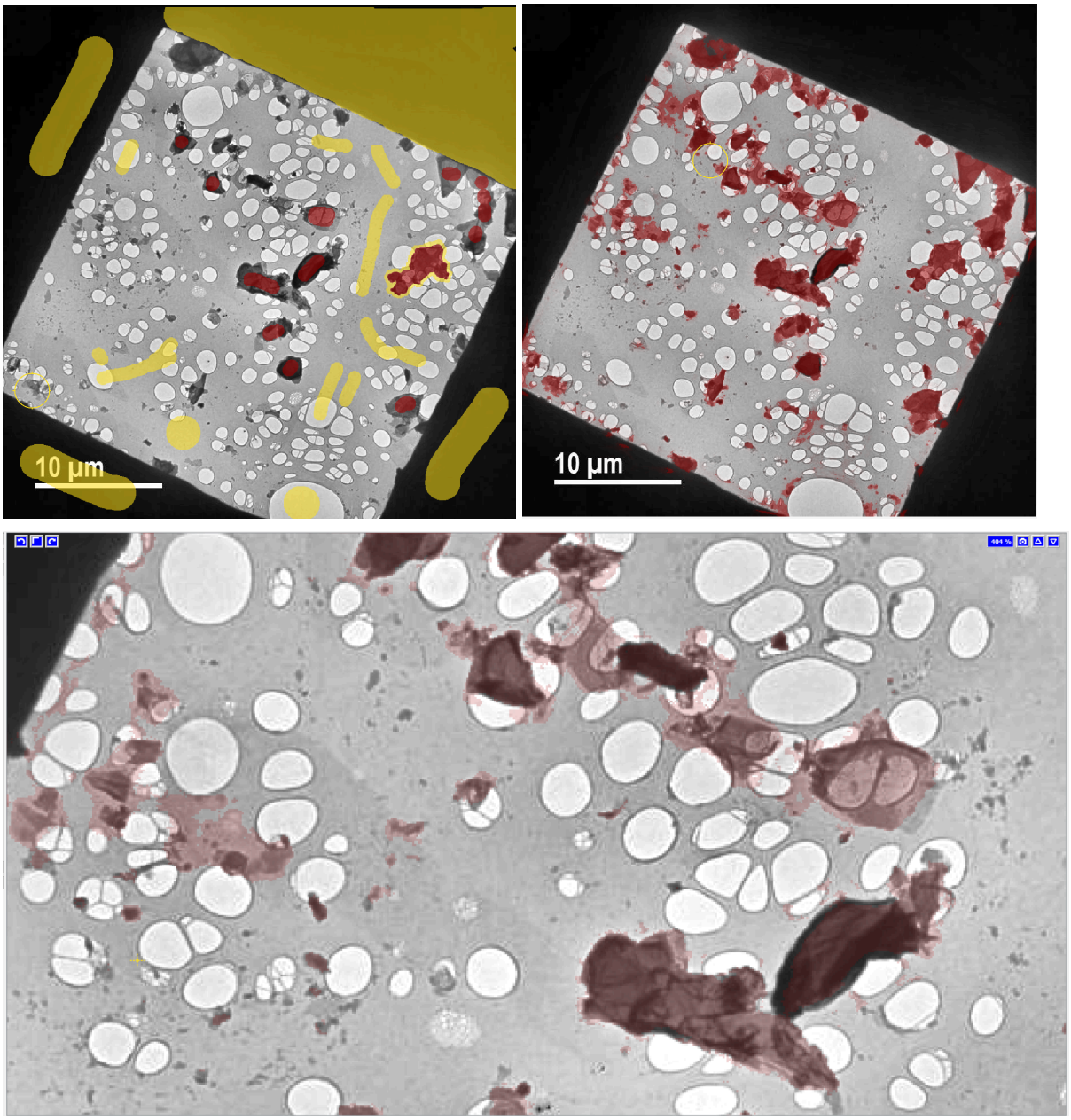

Figure A. Up left: rough manual-labeling in ilastik before training. Up right: output of training RF model after single iteration of manual labeling. Bottom: zoom on output image to see discrepancies.

In the first stage, rough manual labeling was performed in ilastik (Fig A left). Selected structures were annotated together with representative background regions, allowing the classifier to learn object boundaries. Based on this input, a Random Forest model was trained on image-derived features (Fig A right)

Looking at the zoomed crop of the image (Fig A bottom) closer inspection of the results reveals both strengths and limitations of the approach. The method identifies many object boundaries, but the contours are often imprecise: some regions are over-segmented, others under-segmented, and very bright graphene flakes are not detected at all.

Overall, the Random Forest segmentation in ilastik provides a useful preliminary mask generation tool. However, the results require additional fine-tuning and post-processing to achieve the accuracy necessary for subsequent analysis.

A key advantage of ilastik is that the classifier provides per-pixel probability estimates of class membership, allowing to generate pixel class probability maps. These maps were exported in `.h5` format and processed in a dedicated notebook. For instance, in Image B (left) shows the probability distribution for the target class, providing more nuanced information than a binary mask. Separate thresholds can be applied to each image.

The post-processing pipeline converts ilastik probability maps (`.h5`) into binary masks through automated file detection, threshold-based classification, and user-defined configuration of target classes. The results are exported in standardized formats (PNG and NumPy), with organized directory structures to ensure dataset consistency and reproducibility. For quality assurance, overlays and probability heatmaps are generated, enabling visual inspection of threshold adequacy and mask reliability across the dataset.

In *Image B* (right panel), a binary mask is shown after applying the most accurate threshold and exporting the result in binary format. Nevertheless, several deficiencies remain, such as holes within objects or excessive regions being marked (as indicated, for example, by the blue arrow). For this reason, in most cases the binary mask was re-imported into ilastik as a **Label object**, where manual corrections were introduced. Following this refinement step, the updated results were exported again from ilastik in `.h5` format. This procedure is done separately for GFM, carbonhole and undefined class.

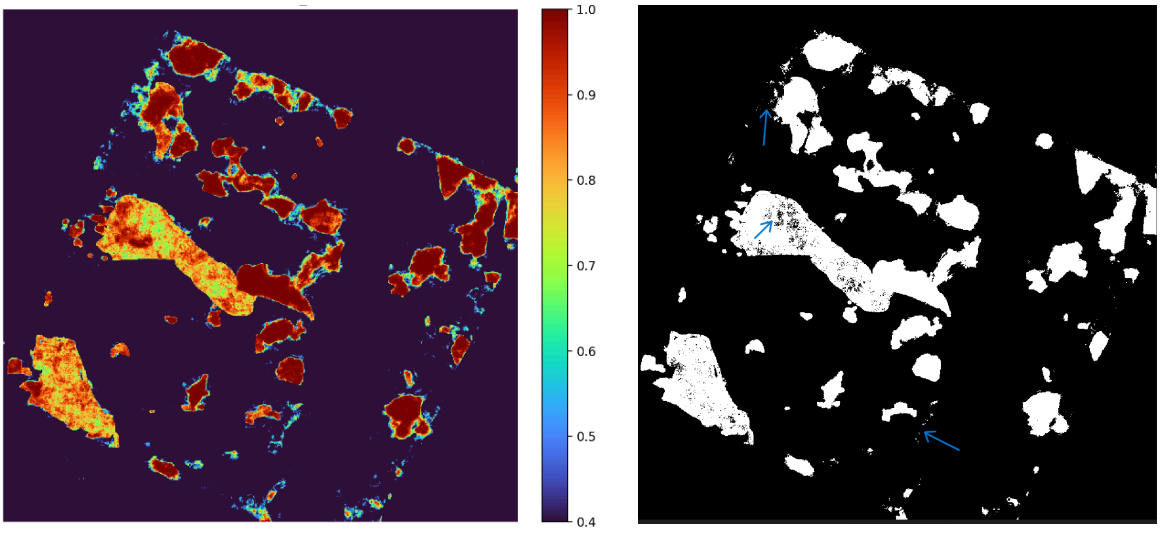

Figure B. Left: Probability map showing the per-pixel likelihood of belonging to the GFM class versus background. Right: Binary mask obtained by applying a 0.7 threshold to the probability map: pixels with a probability greater than 70% of being GFM are set to 1 (white), all others to 0 (black).

Notebook is available here: D_ilastik_h5_to_masks

Post-processing of ilastik label masks for UNet training

Following manual refinement in ilastik, the label annotations were exported as `*_Labels.h5` files. A dedicated post-processing pipeline was developed to transform these annotations into a consistent, UNet-ready dataset containing multi-label masks paired with the corresponding input images.

The pipeline first reads the integer label masks from the ilastik output and harmonizes their spatial resolution (default 512×512) using nearest-neighbor interpolation, thereby preserving class integrity. Nearest-neighbor interpolation was applied during mask resizing. Unlike bilinear or bicubic interpolation, it does not generate intermediate pixel values but directly assigns the class of the closest original pixel. This ensures that categorical labels remain intact and no artificial classes are introduced, thereby preserving class integrity in segmentation masks. Since the refined labels may still contain gaps, unlabeled pixels (class 0) are reassigned to the nearest labeled class using a Euclidean distance transform (EDT), effectively removing isolated voids without creating artificial boundaries. For each void pixel, EDT computes the shortest Euclidean distance to a labeled pixel and assigns its class accordingly. This approach removes isolated gaps while avoiding the introduction of artificial boundaries.

To improve the geometric consistency of the masks, morphological smoothing in the form of binary opening is applied to the target class (default class 2). This operation reduces small edge artifacts while maintaining the overall structure of the objects. Any temporary zeros introduced during reconstruction are reassigned to the background class, ensuring that no spurious unlabeled regions remain.

For each label file, the pipeline automatically identifies the corresponding raw image and guarantees spatial alignment between image and mask. The final dataset consists of non-overlapping, one-hot masks representing the main classes (GFM, carbonhole, undefined) together with their paired images. Quality-control steps include reporting per-channel pixel counts and verifying the absence of overlaps between classes.

The resulting standardized dataset is therefore directly suitable for supervised learning. It emphasizes reproducibility, through fixed target resolution and deterministic operations; cleanliness, through gap filling and morphological refinement; and quality assurance, through systematic checks and summaries. An example of the final masks overlaid on the original images for easier visualization is shown in *Image YY*.

Notebook is available here: E_mask_postprocessing_for_unet

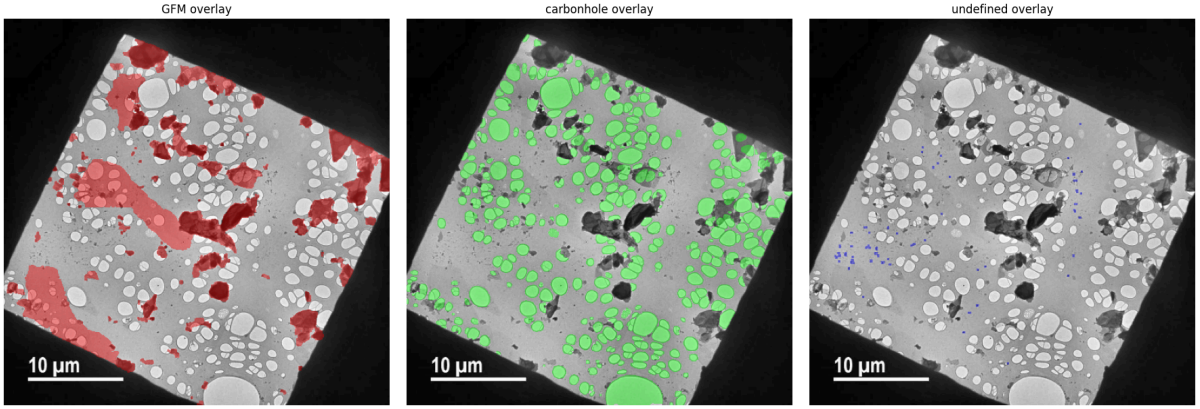

Figure YY. Examples of the final post-processed masks overlaid on the corresponding raw image. The three classes (*GFM -red* , *carbonhole - green*, *undefined - blue*) are shown in distinct colors, illustrating the effect of gap filling and morphological smoothing. This visualization highlights the improved mask consistency and alignment with the underlying structures, facilitating their direct use for supervised segmentation tasks.

4. Deep Learning Approaches

Graphene, with its remarkable physical and chemical properties, is one of the most intensely investigated two-dimensional materials. Transmission electron microscopy (TEM) plays a central role in its characterization, yet manual annotation of TEM images is laborious and subject to bias. Automated segmentation is therefore essential to enable reproducible and scalable quantitative analysis of structural features such as carbon holes, multilayers, and amorphous residues.

Deep learning–based semantic segmentation offers a powerful solution, assigning each pixel to a semantic class. Among the available architectures, U-Net is particularly effective, combining encoder–decoder pathways with skip connections to capture both fine detail and global context. In this study, we adapt U-Net with a pretrained ResNet backbone, leveraging transfer learning to achieve strong performance despite limited training data.

Our dataset consists of grayscale TEM images with manually generated masks, reduced to two principal classes: graphene film and carbon holes. To address class imbalance and data scarcity, we applied geometric and photometric augmentations, alongside patch-based sampling. Training employed class-specific losses: Tversky loss for graphene film, which emphasizes false-positive control, and a hybrid of binary cross-entropy and Dice for carbon holes, which are rare and irregular.

The pipeline integrates mixed-precision training, early stopping, learning-rate scheduling, and threshold tuning. Model performance is reported using IoU, Dice, precision, and recall, with both aggregate metrics and per-image reports. Visual overlays of predictions, ground truth, and error maps provide further interpretability.

This work delivers a reproducible, end-to-end framework for automated segmentation of graphene TEM images. By combining state-of-the-art deep learning techniques with domain-specific strategies, it supports reliable analysis accessible to both materials scientists and non-specialists.

Objectives

We set out to build a **reproducible, data-centric segmentation pipeline** on Vertex AI (PyTorch) that:

1. **Trains a strong U-Net–style model with a modern backbone** (ResNet/EfficientNet/FPN/DeepLab) while keeping inference fast.
2. **Handles imbalance and boundary ambiguity** using tailored losses (Tversky for GFM, BCE+Dice for CH).
3. **Improves generalization** via patch training with overlap, realistic augmentations (including scale jitter), mixed precision (AMP), LR scheduling, and **test-time augmentation (TTA)**.

4. **Evaluates transparently**, with per-class IoU/Dice, per-image tables, threshold tuning on validation, and early stopping on a meaningful target.
5. **Integrates tightly with GCS** for data I/O and (optionally) model artifacts/thresholds.

Dataset Partitioning and Pixel Distribution

A critical step in training segmentation models is the partitioning of data into training, validation, and test sets. This ensures that model performance is assessed not only on the data it learns from but also on previously unseen samples, thereby providing a robust estimate of generalization.

In our study, the dataset of annotated TEM images was divided into three subsets:

- **Training set (≈70%)** — used directly to optimize the model parameters.
- **Validation set (≈20%)** — employed during training to tune hyperparameters, monitor performance, and guide early stopping.
- **Test set (≈10%)** — reserved exclusively for final evaluation, ensuring unbiased assessment of the trained model.

Unlike typical image classification tasks, where balancing is measured at the image level, semantic segmentation requires attention to **pixel-level class distributions**. In TEM images of graphene, this aspect is crucial:

- Pixels corresponding to **graphene film (GFM)** cover extended and heterogeneous regions, often exhibiting complex boundaries.
- Pixels representing **carbon holes (CH)** are abundant and structurally well-defined, typically forming regular circular patterns across many images.
- The **undefined/background** category was merged with GFM to simplify the classification into two principal classes.

To assess the representativeness of each subset, we computed per-pixel counts of GFM and CH across training, validation, and test sets. The resulting bar plots illustrate how pixel counts vary across subsets, providing a diagnostic tool to detect imbalance. Our sanity check explicitly flagged subsets where one of the classes might be underrepresented, ensuring a consistent presence of both GFM and CH across partitions.

This pixel-aware analysis guarantees that both majority and minority regions are represented in every subset. It forms the foundation for subsequent loss design, class reweighting, and threshold tuning strategies used in training.

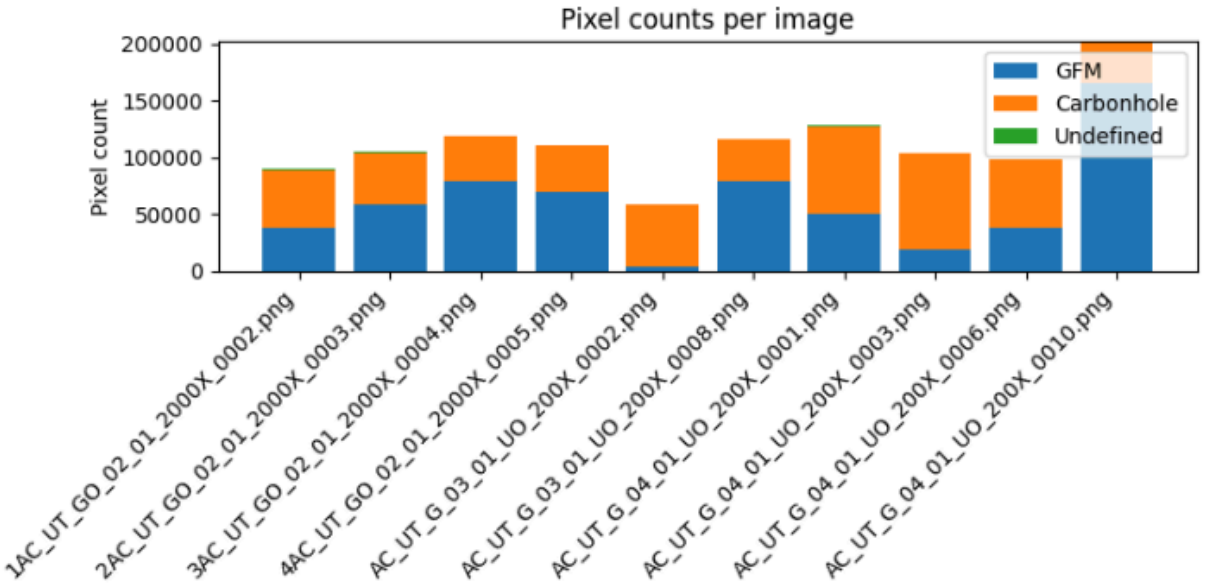

Figure Y. *Pixel-level class distribution per image. Each bar corresponds to a single TEM image, with stacked segments indicating the number of pixels annotated as graphene film (GFM, blue), carbon holes (CH, orange), and undefined/background (green). The plot reveals the inherent non-uniformity of class distributions across the dataset*

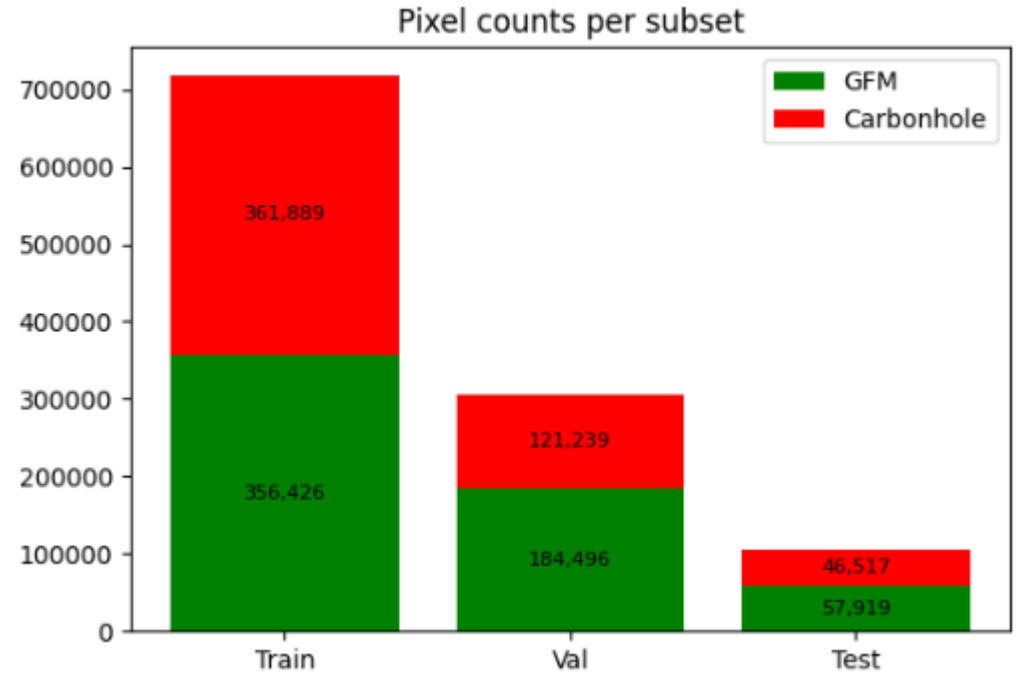

Figure X. *Pixel distribution across dataset partitions (manual split). Bars represent the total number of pixels assigned to graphene film (GFM, green) and carbon holes (CH, red) in the training, validation, and test sets. This visualization ensures that both classes are consistently represented across subsets, reducing the risk of training or evaluation bias.*

Data Augmentation

To improve model robustness and mitigate overfitting, we applied a diverse set of augmentation strategies to the training data. These augmentations simulate variability in experimental conditions, such as differences in imaging contrast, magnification, and acquisition artifacts. Importantly, augmentation was applied **only to the training set**, ensuring that validation and test sets remained representative of real-world evaluation conditions.

The applied augmentations included:

- **Random flips (horizontal and vertical)**: Increase invariance to image orientation.
- **Random 90° rotations**: Further diversify spatial perspectives.
- **Random brightness and contrast adjustments**: Simulate variations in electron beam intensity and sample preparation.
- **Gaussian noise addition**: Mimic sensor or acquisition noise present in TEM images.
- **Random scaling and jittering**: Model magnification differences between 200× and 2000× imaging conditions.
  Gaussian blurring: Simulate the soft focus or drift artifacts sometimes seen in TEM data.

Figure 1 demonstrates these augmentations on a simple synthetic image of a tree. Although not identical to graphene TEM structures, the example illustrates the types of transformations applied.

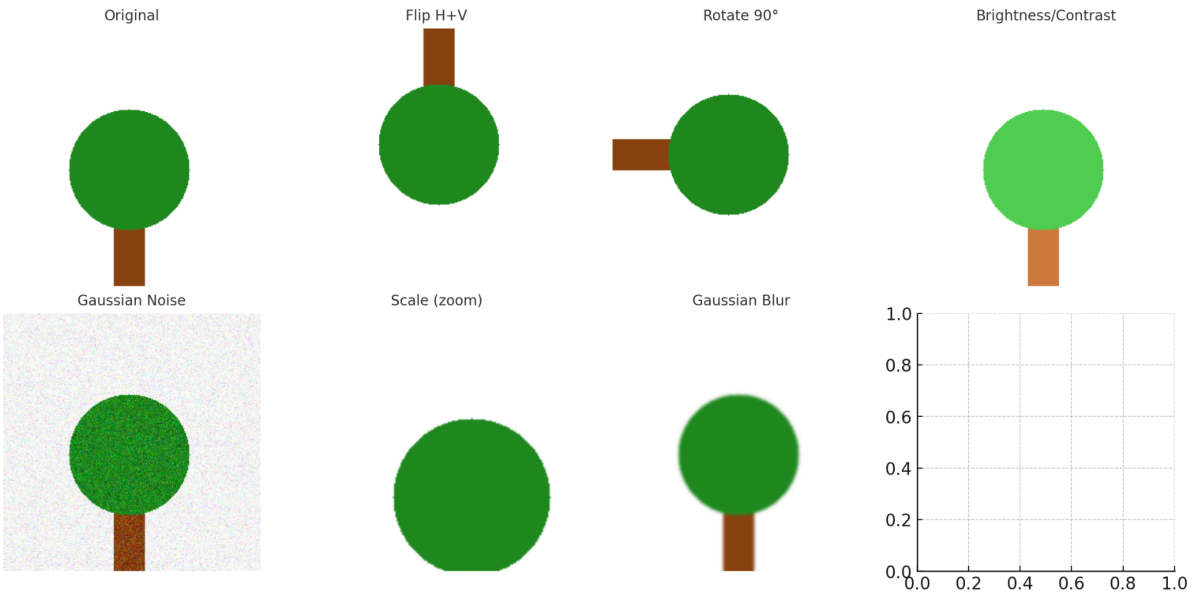

Figure X. *Examples of data augmentation steps applied during training. From left to right: Original, Flip H+V, Rotation 90°, Brightness/Contrast adjustment, Gaussian Noise, Scaling (zoom), Gaussian Blur.*

Additionally, the training strategy employed **patch-based learning**. Instead of processing only entire large TEM micrographs, images were divided into smaller overlapping patches. This approach offers several advantages:

- It reduces GPU memory consumption, enabling training with higher-resolution details.
- It balances the dataset by ensuring that underrepresented classes (e.g., carbon holes vs. GFM regions) appear more frequently in the training batches.
- It improves generalization by forcing the model to learn from localized structures rather than relying on global context only.

In combination, these augmentations and the patch-based strategy helped create a training regime that better reflects the diversity and complexity of TEM graphene imagery.

Patch-Based Cropping and Foreground Filtering

To adapt full-frame TEM micrographs to a form suitable for deep learning, we employed a patch-based training strategy. Each 512×512 image was subdivided into smaller 256×256 windows using a sliding window with a stride of 128 pixels, resulting in 50% overlap between neighboring patches. In principle, this procedure generates nine candidate patches per image.

Since many of these windows contain only background, we implemented a foreground filtering step. Foreground was defined as the union of two annotated classes: graphene flakes and carbon holes. For each candidate patch, the number of pixels belonging to these classes was computed directly from the segmentation masks. Patches were retained only if they contained more than ten foreground pixels, a small threshold chosen to exclude empty or nearly empty windows while preserving those with meaningful structures. This procedure ensured that training focused on content-rich regions, reducing both computational overhead and noise from irrelevant background.

In practice, the filtering step substantially reduced the number of training samples. For the seven annotated training images, the theoretical maximum of 63 candidate patches was reduced to 27 retained patches after foreground filtering. This reduction highlights the strong class imbalance and heterogeneous spatial distribution of features: while graphene and carbon holes are frequent across the dataset, they occupy only limited regions of the micrographs.

The overlapping design of the patching strategy further improved the training dynamics. By allowing border pixels of one patch to appear near the center of another, overlap mitigated edge artifacts and helped the model learn more consistent representations across the field of view. The result is a curated set of image patches that balances efficiency with representativeness, ensuring that the network is trained on regions where meaningful structures are present.

Model Architecture

The segmentation model employed in this study is a **U-Net architecture** with a **ResNet-34 encoder backbone**. U-Net is a widely used convolutional neural network (CNN) architecture designed for biomedical and materials imaging, characterized by its symmetric encoder–decoder structure with skip connections. The encoder progressively reduces spatial resolution while increasing the number of feature channels, enabling the extraction of hierarchical features. The decoder reconstructs the spatial resolution by combining upsampled features with corresponding encoder activations, thus preserving fine-grained details essential for precise segmentation of nanostructures.

In our implementation, the encoder consists of a **ResNet-34** backbone pre-initialized with weights trained on ImageNet. It begins with a 7×7 convolution and max-pooling layer, followed by four stages of residual blocks. The number of channels increases from 64 to 512 across these stages, with strided convolutions reducing spatial resolution at each step. This deep residual encoder enables the extraction of high-level semantic features while maintaining stable gradient propagation.

The decoder mirrors the encoder with a sequence of **five decoding blocks**, each consisting of two convolutional layers with batch normalization and ReLU activations. At every stage, features from the encoder are concatenated with the upsampled decoder features through **skip connections**, which preserve fine-grained spatial information. The final segmentation head uses a 3×3 convolution to map decoder features to **two output channels** (graphene flakes and carbon holes), producing dense pixel-level predictions.

This design offers several advantages:

- **ResNet backbone** ensures robust feature extraction and stable optimization.(might be changed to other pre-trained model)
- **Skip connections** mitigate information loss during downsampling, crucial for accurately delineating small-scale structures such as graphene edges.
- **Two-channel output** is well suited for binary multilabel segmentation, where overlapping classes may exist.

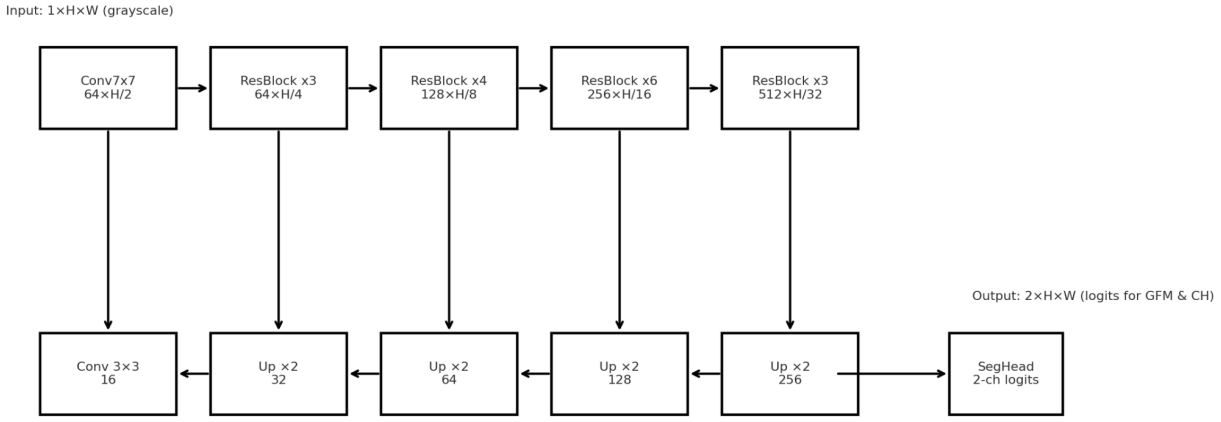

Figure X. *Neural network architecture. It shows the ResNet-34 encoder (Conv7×7, then residual stages 64/128/256/512 with downsampling), the U-Net decoder (four upsampling blocks + final 3×3 conv), and the skip connections from each encoder stage to its corresponding decoder block.*
Input: 1×H×W grayscale. Output: 2×H×W logits (GFM & CH).

Loss Functions and Training Techniques

Used techniques summary:

- **Losses**: Tversky for GFM, BCE+Dice for Carbonhole.
- **Optimizer**: AdamW.
- **Scheduler**: ReduceLROnPlateau (on mean IoU, not cosine annealing).
- **Mixed precision training**: AMP (fp16/bf16) with gradient scaling.
- **Early stopping**: based on mean IoU.
- **Threshold tuning**: grid + refinement search for best per-class sigmoid thresholds.

Training segmentation models on TEM images of graphene requires special care: the structures of interest (graphene flakes, carbon support holes) vary in size, regularity, and frequency, and class imbalance is severe. For example, carbon holes are abundant and well-defined, while graphene flakes can be thin, irregular, or partially overlapping. Below we describe each method in this project, explaining both its mathematical definition and why it is particularly suited for this domain.

Intersection-over-Union (IoU)

Accuracy alone is misleading when the background dominates (most pixels are background). IoU directly measures overlap for the minority structures (GFM, CH), ensuring that a predicted flake or hole is really in the right place and not just guessing background.

Dice Loss

$$Dice = 2TP/(2TP + FP + FN)$$

Graphene flakes are often small and filament-like, occupying only a fraction of the image. Dice loss gives more weight to such rare, thin structures and prevents them from being ignored during training (a common issue with pixel-wise BCE alone).

Tversky Loss

$$Tversky = TP/(TP + \alpha FP + \beta FN)$$

For graphene flake masks (GFM), false positives (predicting graphene where none exists) are especially problematic, since background carbon film or contamination can look similar. Using α=0.8,β=0.2\alpha=0.8, \beta=0.2α=0.8,β=0.2 strongly penalizes FP, ensuring the model does not “hallucinate” graphene in clean carbon areas.

BCE + Dice Combination

BCE ensures per-pixel accuracy, Dice ensures overlap quality. For carbon holes (CH), the structures are large, circular, and abundant. BCE enforces accurate hole boundaries at the pixel level, while Dice ensures correct area coverage. The combination captures both global shape and pixel-wise precision.

Early Stopping

Training halts when validation mean IoU does not improve for 10 epochs.
TEM datasets are small and prone to overfitting. Without early stopping, the model could memorize training holes or flakes, losing generalization to new samples. Early stopping ensures the model captures transferable patterns (e.g., generic hole geometry, graphene edge textures).

Mixed Precision Training (AMP)

Training in fp16/bf16 precision with automatic gradient scaling.
TEM images are high-resolution; training with full precision quickly exhausts GPU memory. AMP allows larger batch sizes and faster iterations, making it possible to process realistic crops without downsampling too aggressively (which could erase small flakes).

Learning Rate Scheduling

ReduceLROnPlateau reduces learning rate when IoU stagnates. Graphene segmentation is challenging: the model often learns easy classes first (carbon holes), then struggles with subtle GFM. The scheduler allows coarse updates initially but switches to fine adjustments once the easy patterns are learned, enabling better convergence on the difficult graphene structures.

Threshold Tuning

Optimal binarization thresholds are searched for each class separately. The optimal probability threshold differs between classes:

- Graphene flakes are thin and low-contrast — requiring a lower threshold to avoid missing faint edges.
- Carbon holes are large and clear — requiring a higher threshold to avoid false positives from background gaps.

By tuning thresholds automatically, the model respects these physical differences.

Why These Techniques Are Superior to "Standard" Training

Standard approach: Use BCE or Dice alone, default 0.5 threshold, fixed learning rate, and no early stopping.
Problem: This would bias the model toward background, overestimate graphene, and ignore subtle edges.

- Improved approach (ours):
    - Custom losses (Tversky for GFM, BCE+Dice for CH) → explicitly match class-specific needs.
    - Threshold tuning → adapts to faint graphene vs clear holes.
    - Scheduler & AMP → stable, efficient training without losing small features.
    - Early stopping → prevents memorization of a tiny dataset.

Together, these choices reflect not just "best practices" in ML but a tailored pipeline that matches the physical nature of TEM graphene images: rare flakes, abundant circular holes, strong class imbalance, and subtle edge transitions.

Training progress is summarized in Figure X. The training loss decreased steadily over the first 15 epochs, accompanied by a rapid increase in validation IoU. After approximately epoch 25, the validation performance plateaued, indicating convergence. Early stopping was triggered after 34 epochs, preventing overfitting. The cosine-like reduction in learning rate helped stabilize convergence in the later stages of training.

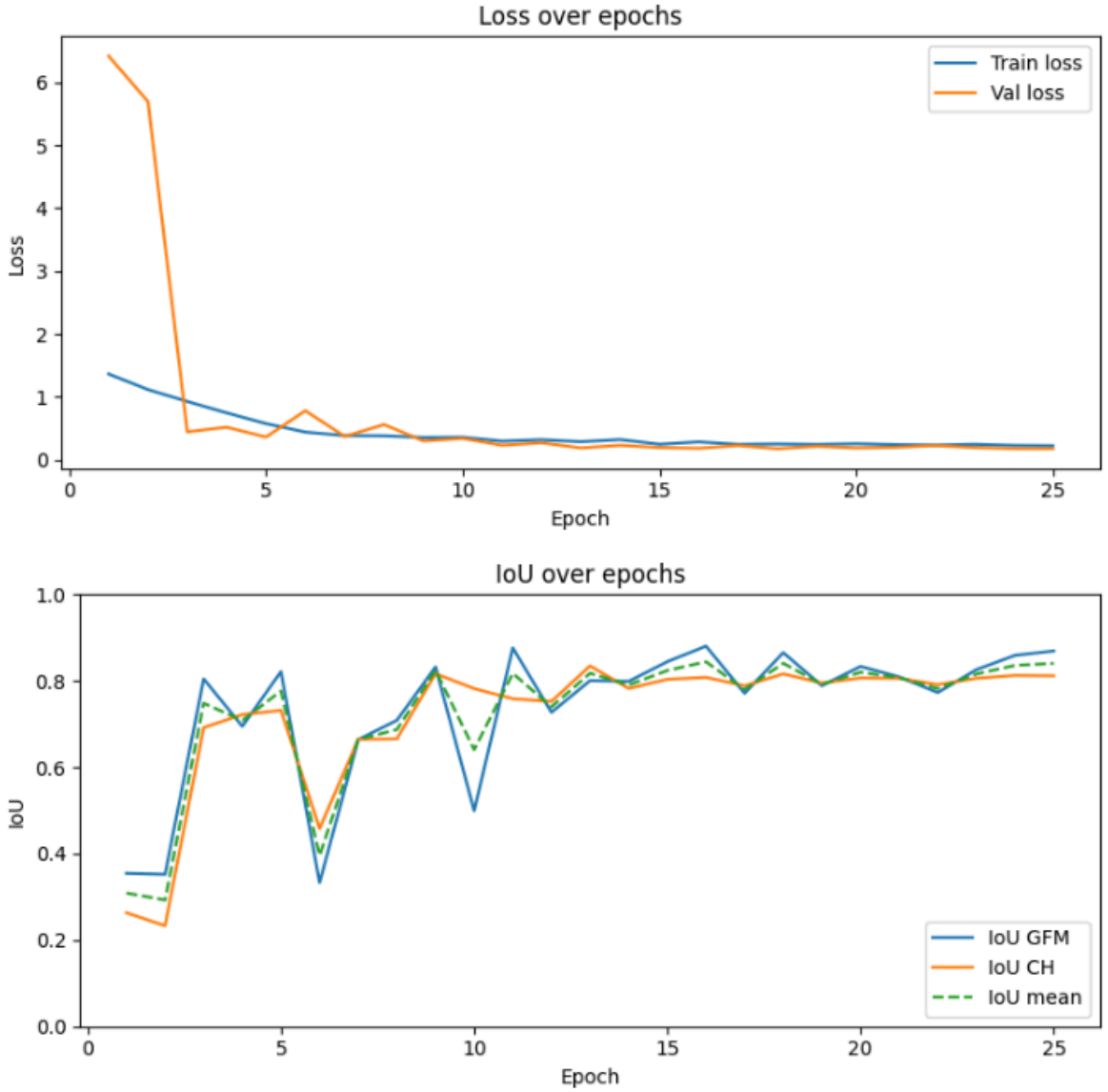

Figure X. Training progress of the U-Net with ResNet backbone. (Top) Training and validation loss over 25 epochs. Both curves show rapid convergence within the first 5 epochs, followed by stable minimization. (Bottom) Intersection-over-Union (IoU) evolution for GFM and carbon holes (CH), with the dashed line showing the mean IoU. Performance stabilized above 0.8 IoU after ~15 epochs, indicating robust segmentation capability without signs of overfitting.

Loss Functions and Training Techniques

The quality of segmentation was assessed using standard pixel-level metrics widely applied in image analysis:

- **Intersection-over-Union (IoU):**

$$IoU = TP/(TP + FP + FN)$$

where TP are true positives, FP false positives, and FN false negatives. IoU quantifies the overlap between predicted and ground truth masks, with 1.0 indicating perfect agreement.

- **Dice coefficient (F1 score for segmentation):**

$$Dice = 2TP/(2TP + FP + FN)$$

Dice emphasizes correctly predicted pixels, especially for smaller structures, and is closely related to IoU.

- **Precision**

$$precision = TP/(TP + FP)$$

the fraction of predicted positive pixels that are correct. High precision means few false positives.

- **Recall (Sensitivity):**

$$recall = TP/(TP + FN)$$

the fraction of ground truth positive pixels that are correctly detected. High recall means few false negatives.

- **Specificity:**

$$specificity = TN/(TN + FP)$$

the ability to correctly classify background pixels.

After threshold tuning on the validation set, the model achieved the following results:

- Validation set
    - GFM: IoU = 0.823, Dice = 0.903, Precision = 0.874, Recall = 0.934
    - Carbonhole (CH): IoU = 0.812, Dice = 0.896, Precision = 0.861, Recall = 0.938
    - Mean IoU (GFM+CH): 0.818
- Test set
    - GFM: IoU = 0.808, Dice = 0.893, Precision = 0.849, Recall = 0.941
    - Carbonhole (CH): IoU = 0.803, Dice = 0.891, Precision = 0.856, Recall = 0.926
    - Mean IoU: 0.806

These results demonstrate that the model generalizes well from the validation to the test set, with mean IoU consistently above 0.80. The recall values for both classes exceed 0.92, highlighting the robustness of the model in capturing graphene flakes and carbon holes, even at the cost of a slight drop in precision.

Table 1. Validation set performance.

Class	*IoU*	*Dice*	*Precision*	*Recall*	*Specificity*
GFM	*0.823*	*0.903*	*0.874*	*0.934*	*0.962*
Carbonhole	*0.812*	*0.896*	*0.861*	*0.938*	*0.957*
Mean	*0.818*	*0.899*	*0.868*	*0.936*	*0.960*

Table 2. Test set performance.

Class	*IoU*	*Dice*	*Precision*	*Recall*	*Specificity*
GFM	*0.808*	*0.893*	*0.849*	*0.941*	*0.955*
Carbonhole	*0.803*	*0.891*	*0.856*	*0.926*	*0.951*
Mean	*0.806*	*0.892*	*0.853*	*0.934*	*0.953*

- Both **validation and test results** confirm high segmentation quality, with **mean IoU > 0.80** and **Dice ≈ 0.90**.

- **Recall values > 0.92** across classes indicate that the model is particularly effective at detecting relevant structures, minimizing missed pixels.
- **Precision is slightly lower**, especially for GFM, due to the higher variability of graphene flake morphology compared to the highly regular circular holes.
- The close agreement between validation and test performance demonstrates that the model generalizes well, with no sign of overfitting.

The visualisation can be found in Figure X. The model predictions closely follow the annotated structures, with particularly precise delineation of circular carbon holes and reasonable capture of irregular graphene flakes. Tre where none was present). Blue pixels = false negatives (missed regions of ground truth).
The majority of pixels appear in green, confirming overall high accuracy. Residual errors are mainly false positives on the GFM class, where the model occasionally confuses background texture with thin graphene flakes. Looking on error map, The majority of pixels appear in green, confirming overall high accuracy. Residual errors are mainly false positives on the GFM class, where the model occasionally confuses background texture with thin graphene flakes.

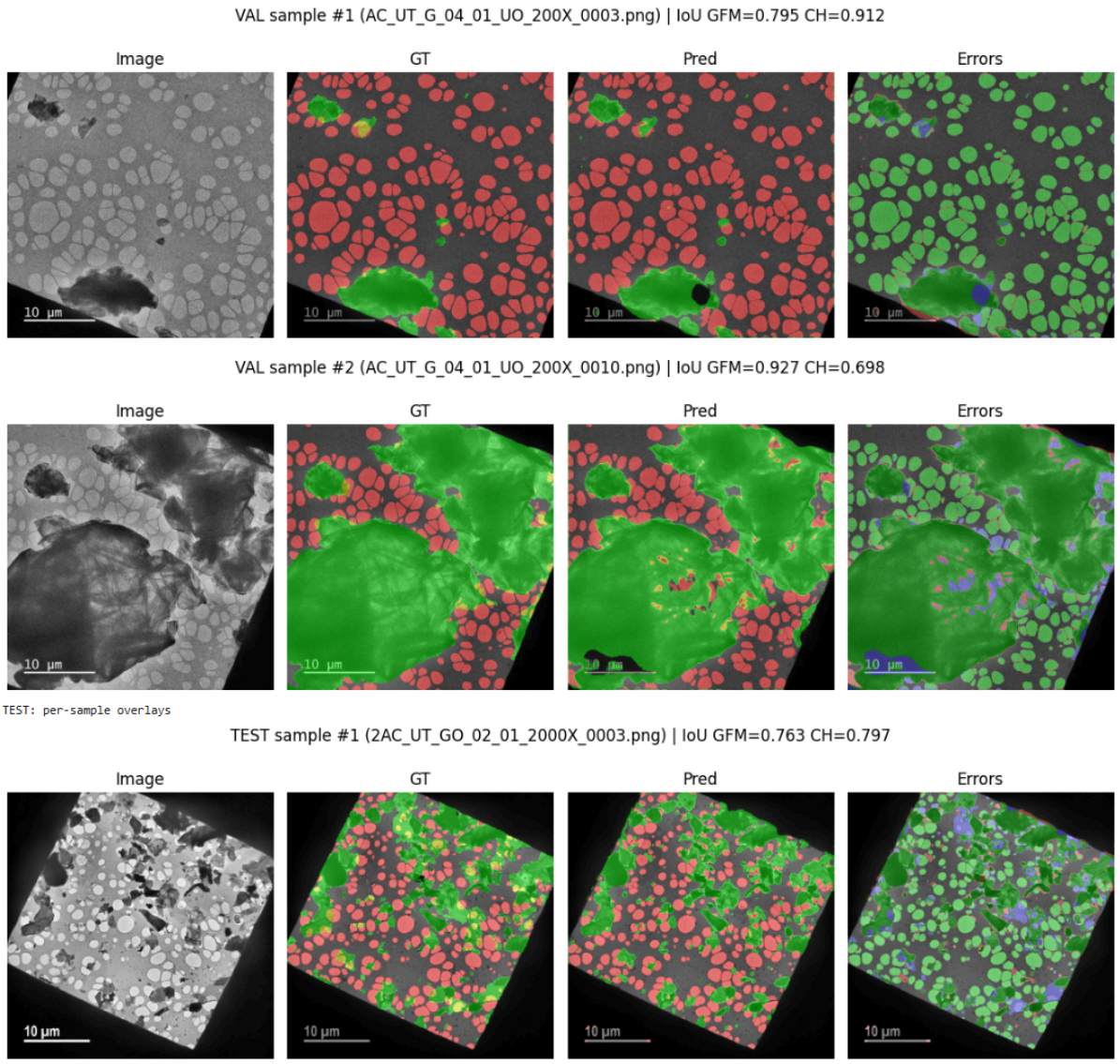

Figure X. Example segmentation overlays on validation samples. Each row shows: the raw TEM input (left), ground-truth annotation (middle), and model prediction (right). Green corresponds to graphene flakes and multilayers (GFM). Red corresponds to carbon holes (CH). Yellow corresponds to regions where both GFM and CH overlap. The last image is Error maps for selected images: overlay comparison between predictions and ground truth. Green pixels = true positives (correctly segmented). Red pixels = false positives (model predicted structure where none was present). Blue pixels = false negatives (missed regions of ground truth).

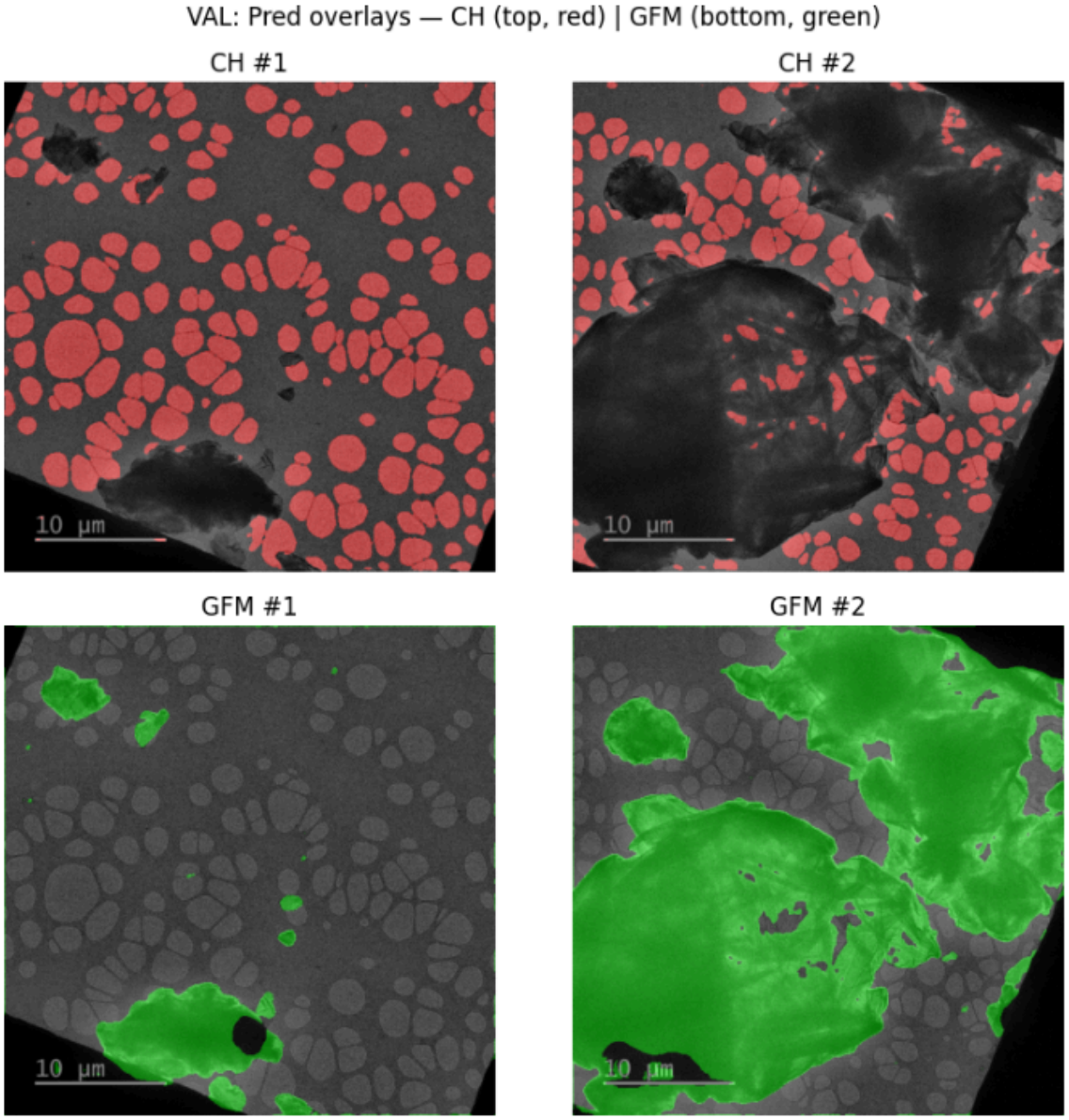

Figure Y. Prediction overlays for carbonholes on validation dataset.

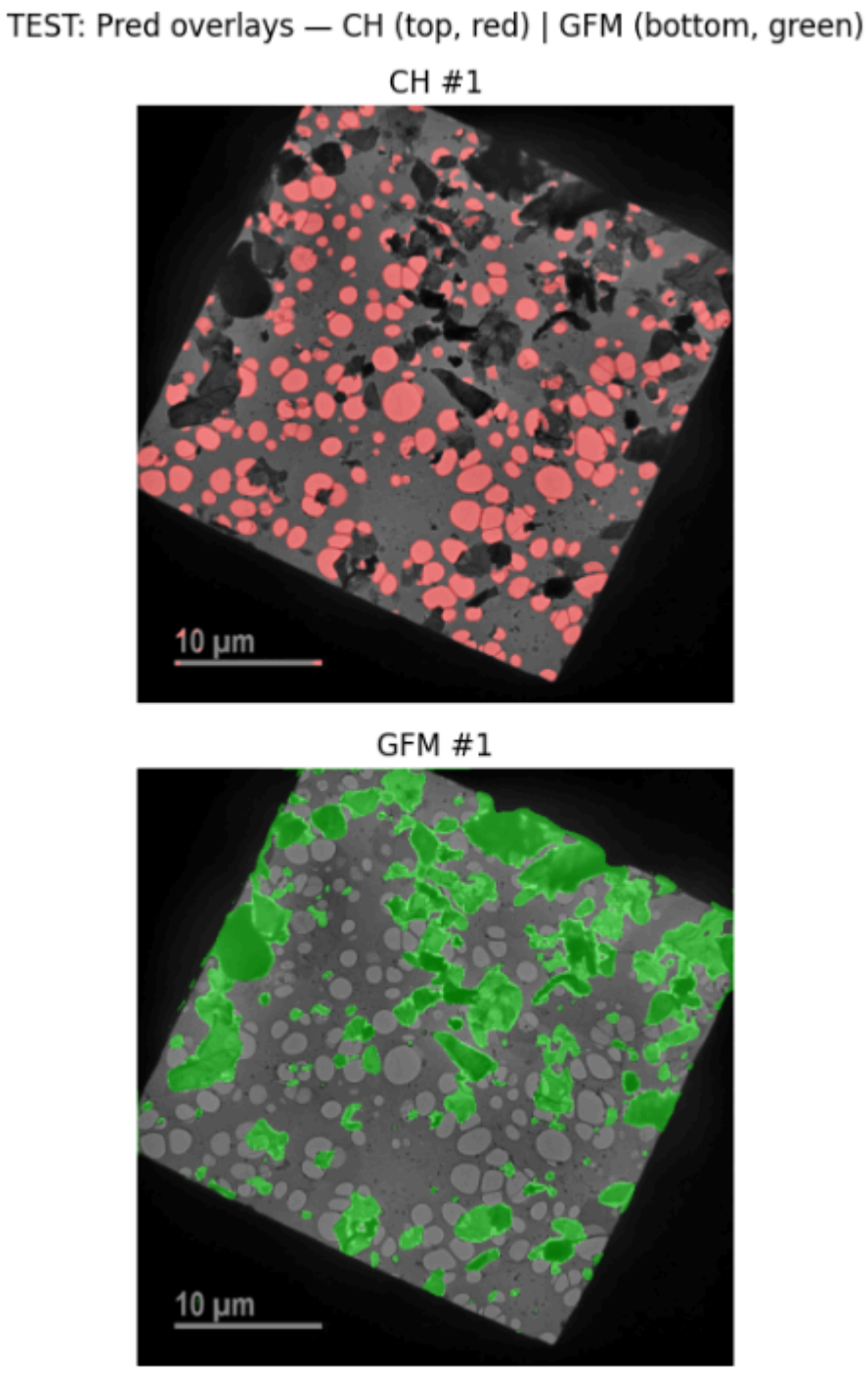

Figure Y. Prediction overlays for carbonholes on test dataset.

5. Future Work

While the present U-Net with ResNet backbone demonstrates strong performance in segmenting graphene flakes and carbon holes, several avenues exist for improvement:

- **Dataset Expansion:** Increasing the number and diversity of annotated TEM images will improve generalization across different acquisition conditions.

- **Advanced Architectures**: Testing transformer-based backbones (e.g., Swin-UNet, SegFormer) or hybrid CNN–transformer models may better capture long-range dependencies in graphene morphology.

- **Class Refinement:** Introducing finer class labels (e.g., monolayer vs. multilayer graphene, folded edges, contamination) could enable more detailed characterization.

- **Uncertainty Estimation**: Incorporating Bayesian dropout or ensemble methods would quantify prediction confidence, which is particularly important for guiding experimental decisions.

- **Integration with Automated Workflows**: Linking segmentation outputs with downstream analysis (particle size distribution, porosity, defect density) would make the tool directly usable in materials science pipelines.

This roadmap balances technical improvements with practical integration, aiming to evolve the model from a proof-of-concept towards a production-ready tool for graphene TEM analysis.