



**Instituto Superior
de Engenharia**

Politécnico de Coimbra

DEPARTMENT OF ELECTRICAL ENGINEERING

Cell Segmentation in FIB-SEM Microscopy Images

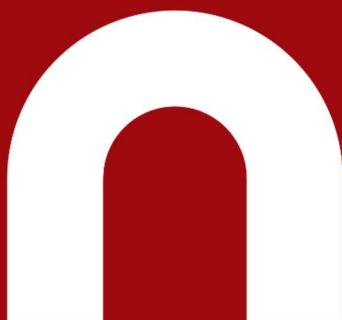
Project Report to fulfill the Master's degree in Electrical Engineering
Branch of Automation and Communications in Energy Systems

Author

João Ricardo Neves Menezes de Lemos

Supervisor

Doutor Fernando José Pimentel Lopes



INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, January 2026

RESUMO

Este Trabalho de Projeto investiga o impacto da compressão de imagens na performance de modelos de segmentação baseados em Deep Learning aplicados a imagens FIB-SEM (Focused Ion Beam Scanning Electron Microscopy). Modelos de YOLO foram treinados e avaliados através várias configurações experimentais, com consideração cuidadosa do manuseamento do dataset, da metodologia de treino, e da configuração da inferência. Experiências iniciais destacaram a importância da separação estrita do dataset FIB-SEM para avaliação realista da generalização.

O efeito da compressão foi avaliado aplicando um modelo de JPEG AI pré-treinado às imagens de teste, alcançando elevadas taxas de compressão e métricas objetivas favoráveis, demonstrando o potencial de compressão baseada em deep-learning para armazenamento e transmissão de dados de microscopia de alta resolução. O desempenho da segmentação em imagens comprimidas mostrou uma redução gradual nas métricas de qualidade da segmentação com o aumento das taxas de compressão, indicando que o modelo de compressão estava a suprimir alguma informação relevante para a segmentação ao reduzir o número de bits da representação comprimida.

Os resultados demonstram que sistemas de detecção/segmentação baseados em Deep Learning podem operar efetivamente em imagens comprimidas, apoiando o uso de abordagens conscientes em compressão para análise eficiente de imagens médicas.

Palavras-chave: FIB-SEM, YOLO, JPEG AI, segmentação de imagem, compressão de imagem

ABSTRACT

This Master's project investigates the impact of image compression on the performance of Deep Learning-based segmentation models applied to FIB-SEM (Focused Ion Beam Scanning Electron Microscopy) imagery. YOLO models were trained and evaluated under various experimental setups, with careful consideration for dataset handling, training methodology, and inference configuration. Initial experiments highlighted the importance of strict FIB-SEM dataset separation for realistic generalization assessment.

The effect of compression was evaluated by applying a pre-trained JPEG AI model to test images, achieving high compression ratios and favorable objective metrics, demonstrating the potential of learned compression for storage and transmission of high-resolution microscopy data. Segmentation performance on compressed images showed a gradual reduction on segmentation quality metrics with the compressed bit rates, indicating that the compression model was suppressing some segmentation-relevant information when reducing bit rate.

The results demonstrate that deep learning-based detection/segmentation systems can operate effectively on learning-based compressed images, when compression ratio is controlled and compression effects are monitored, supporting the use of compression-aware approaches for efficient medical image analysis.

Keywords:

FIB-SEM, YOLO, JPEG AI, image segmentation, image compression

EPÍGRAFE

"O começo de todas as ciências é o espanto de as coisas serem o que são."

Aristóteles

DEDICATÓRIA

Dedico este trabalho a todos os que me apoiaram da sua maneira, e nunca me deixaram desistir.

AGRADECIMENTOS

Agradeço o apoio a todos que me ajudaram a realizar este projeto. Agradeço aos meus pais e restante família por serem pessoas com quem poderia contar em momentos menos que ótimos. Agradeço aos meus amigos por todo o apoio e direção durante a realização desta etapa. Agradeço também ao meu orientador Professor Fernando Lopes por me orientar e ser paciente comigo. Agradeço aos investigadores do projeto CIBME pelo material disponibilizado utilizado no desenvolvimento do trabalho, especialmente ao Francisco Cunha por me ajudar a resolver problemas com o respetivo material. Agradeço ao Professor Luis Cruz pelo apoio no âmbito do JPEG-AI e da organização dos recursos necessários para a realização do projeto. Agradeço também aos demais colegas de laboratório no Instituto de Telecomunicações, em especial ao Pedro Rocha, por me orientarem na resolução de problemas e validação de resultados.

INDEX

Resumo	i
Abstract	ii
Epígrafe	iii
Dedicatória	iv
Agradecimentos	v
Index	vi
Table Index	viii
Figure Index	x
1 Introduction	1
2 State of the Art	3
2.1 FIB-SEM Imaging	3
2.2 Image Segmentation Methods	4
2.2.1 Classic Segmentation	4
2.2.2 Deep-Learning Based Segmentation	6
2.3 Image Compression Methods	9
2.3.1 Lossless Image Compression Methods	9
2.3.2 Lossy Image Compression Methods	11
2.3.3 Learning-Based Image Compression	12
3 Dataset	15
4 Image Segmentation	21
4.1 YOLO Detection and Fundamentals	22
4.2 YOLO Segmentation with Data Augmentation and Multi-Plane Sampling	30
4.3 YOLO Segmentation with Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	41

Cell Segmentation in FIB-SEM Microscopy Images

5	Image Compression with JPEG AI	55
5.1	Overview of the coding software	55
5.2	Quantitative evaluation metrics	57
5.3	Preparation of test images	61
5.4	Compression results	61
6	Compressed image segmentation	65
7	Conclusion	70
	Bibliographic References	72
	Annexes	77
	Annex A - Teledyne Dalsa's Sherlock Usage	78
	Annex B - Compression Results for Base operation point and tools off - GPU- LH mode	79
	Annex C - Compression Results for Base operation point and only ResVarScale mode	81
	Annex D - Compression Results for Base operation point and only ICCI mode	83
	Annex E - Compression Results for Base operation point and only LSBS mode	85
	Annex F - Compression Results for Base operation point and only ECThread8 mode	87
	Annex G - Compression Results for High operation point and tools of - GPU mode	89
	Annex H - Compression Results for High operation point and tools of - GPU- LH mode	91
	Annex I - Compression Results for High operation point and only ResVarScale mode	93
	Annex J - Compression Results for High operation point and only ICCI mode	95
	Annex K - Compression Results for High operation point and only LSBS mode	97
	Annex L - Compression Results for High operation point and only ECThread8 mode	99

TABLE INDEX

4.1	Difference among the detection models of YOLOv8 [46] for the COCO set	25
4.2	Test set results for confidence threshold of 0,5 of Detection Task	29
4.3	Absolute and relative count of each instance of each class	34
4.4	Difference among the segmentation models of YOLOv8 [46] on COCO set	35
4.5	Test set results with confidence threshold of 0,5 for Data Augmentation, Multi-Plane Sampling Segmentation Task	39
4.6	Dataset separation into YOLOv8 sets	42
4.7	Hyper-parameter Tuning Settings	43
4.8	Best models for the first combination of variables	43
4.9	Best model for the second combination of variables	44
4.10	Best model for the third combination of variables	44
4.11	Best model for the fourth combination of variables	44
4.12	Metrics of the best model configuration obtained through Hyper-parameter tuning	44
4.13	YOLO augmentation arguments values and functions	45
4.14	Test set results with confidence threshold of 0,2 for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	49
4.15	Class-wise Precision Results	53
4.16	Class-wise Recall Results	53
4.17	Class-wise F1-Score Results	53
5.1	Evaluated compression metrics and limiting values	60
5.2	Average values of metrics for Base operation point on tools off - GPU compression	62
5.3	Standard Deviation values of metrics for Base operation point on tools off - GPU compression	63
6.1	Quantitative results from applying the segmentation model on base op- eration point on tools off GPU compression mode	66
7.1	Average values of metrics for Base operation point on tools off - GPU-LH compression	79
7.2	Average values of metrics for Base operation point on only ResVarScale compression	81

Cell Segmentation in FIB-SEM Microscopy Images

7.3	Average values of metrics for Base operation point on only ICCI compression	83
7.4	Average values of metrics for Base operation point on only LSBS compression	85
7.5	Average values of metrics for Base operation point on only ECThread8 compression	87
7.6	Average values of metrics for High operation point on tools off - GPU compression	89
7.7	Average values of metrics for High operation point on tools off - GPU-LH compression	91
7.8	Average values of metrics for High operation point on only ResVarScale compression	93
7.9	Average values of metrics for High operation point on only ICCI compression	95
7.10	Average values of metrics for High operation point on only LSBS compression	97
7.11	Average values of metrics for High operation point on only ECThread8 compression	99

FIGURE INDEX

1.1	Summary of the elapsed procedure	2
2.1	FIB-SEM Image extraction [7]	4
2.2	Example of a FIB-SEM image [9]	5
2.3	Histogram of a typical image from the dataset provided for this project .	5
2.4	Applying K-means clustering to the image in Figure 2.3	6
2.5	U-Net Architecture [12]	7
2.6	3D U-Net Architecture [14]	8
2.7	General workings of Image Compression	10
2.8	Huffman Coding	10
2.9	Generic representation of an Autoencoder	12
2.10	JPEG AI Framework [30]	14
3.1	Three planes of the volume intercepted by the center point: (a) is extracted from a top perspective, referred as Plane Y; (b) is extracted from a side perspective, referred as Plane X; (c) is extracted from a front facing perspective, equating to the original set and referred as Plane Z; and (d) represents how the perspectives were obtained from the volume. . .	15
3.2	Image extraction model with resource to FIB-SEM technology [33] . . .	16
3.3	Appearance of the TIFF file with the images captured by FIB-SEM technology	17
3.4	Visual difference between 16 bit per pixel images (left) and 8 bit per pixel images (right)	17
3.5	Numerical variance among the four studied classes	18
3.6	Visual contrast among the organelles and their respective location as indicated by the corresponding mask	19
4.1	Representation of the file tree of the YOLOv8 Dataset	22
4.2	Normalised Confusion Matrix of Validation of Detection Task	27
4.3	Interpretation of a Confusion Matrix and usage of presented values for Precision, Recall, Accuracy and F1-Score	28
4.4	F1-Confidence Curve of Validation of Detection Task	28
4.5	Precision-Confidence Curve of Validation of Detection Task	29
4.6	Detection of organelles done by YOLOv8 in Detection Task	30

4.7	Example of an image sliced into 3 512x512 images	32
4.8	Extremities of the volume when analysed from the X plane ((a) and (b)) and from the Y plane ((c) and (d))	33
4.9	Augmentations performed on the images and masks	35
4.10	F1-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task	36
4.11	Precision-Confidence Curve for Validation for Data Augmentation, Multi- Plane Sampling Segmentation Task	37
4.12	Recall-Confidence Curve for Validation for Data Augmentation, Multi- Plane Sampling Segmentation Task	38
4.13	Precision-Recall Curve for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task	38
4.14	Normalised Confusion Matrix for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task	39
4.15	Detection of organelles done by YOLOv8 for Data Augmentation, Multi- Plane Sampling Segmentation Task	40
4.16	F1-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	46
4.17	Precision-Confidence Curve for Validation for Data Augmentation, Multi- Plane Sampling and Hyper-parameter Tuning	47
4.18	Recall-Confidence Curve for Validation for Data Augmentation, Multi- Plane Sampling and Hyper-parameter Tuning	47
4.19	Precision-Recall Curve for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	48
4.20	Normalised Confusion Matrix for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	48
4.21	Detection of organelles done by YOLOv8 for Data Augmentation, Multi- Plane Sampling and Hyper-parameter Tuning	49
4.22	F1-Confidence Curve for Test for Data Augmentation, Multi-Plane Sam- pling and Hyper-parameter Tuning	50
4.23	Precision-Confidence Curve for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	50
4.24	Recall-Confidence Curve for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning	51
4.25	Precision-Recall Curve for Test for Data Augmentation, Multi-Plane Sam- pling and Hyper-parameter Tuning	52
4.26	Normalised Confusion Matrix for Test for Data Augmentation, Multi- Plane Sampling and Hyper-parameter Tuning	52
4.27	Segmentation of organelles done by YOLOv8 and YOLO26	54

5.1	Reconstructed image of Base operation point on tools off - GPU compression at 6% bit rate	63
5.2	Reconstructed image of Base operation point on tools off - GPU compression at 75% bit rate	64
6.1	Evolution of Segmentation Metrics means for all classes with Compression Rates	67
6.2	Segmentation metrics of each class according to compression stage . . .	67
6.3	Predictions of the yolov8m segmentation model on compressed Images .	69
7.1	Example of working Sherlock for ROI information extraction	78
7.2	Reconstructed image of Base operation point on tools off – GPU LH compression at 6% bit rate	80
7.3	Reconstructed image of Base operation point on tools off – GPU LH compression at 75% bit rate	80
7.4	Reconstructed image of Base operation point on only ResVarScale compression at 6% bit rate	82
7.5	Reconstructed image of Base operation point on only ResVarScale compression at 75% bit rate	82
7.6	Reconstructed image of Base operation point on only ICCI compression at 6% bit rate	84
7.7	Reconstructed image of Base operation point on only ICCI compression at 75% bit rate	84
7.8	Reconstructed image of Base operation point on only LSBS compression at 6% bit rate	86
7.9	Reconstructed image of Base operation point on only LSBS compression at 75% bit rate	86
7.10	Reconstructed image of Base operation point on only ECThread8 compression at 6% bit rate	88
7.11	Reconstructed image of Base operation point on only ECThread8 compression at 75% bit rate	88
7.12	Reconstructed image of High operation point on tools off - GPU compression at 6% bit rate	90
7.13	Reconstructed image of Base operation point on tools off - GPU compression at 75% bit rate	90
7.14	Reconstructed image of High operation point on tools off – GPU LH compression at 6% bit rate	92
7.15	Reconstructed image of High operation point on tools off – GPU LH compression at 75% bit rate	92
7.16	Reconstructed image of High operation point on only ResVarScale compression at 6% bit rate	94

Cell Segmentation in FIB-SEM Microscopy Images

7.17	Reconstructed image of High operation point on only ResVarScale compression at 75% bit rate	94
7.18	Reconstructed image of High operation point on only ICCI compression at 6% bit rate	96
7.19	Reconstructed image of High operation point on only ICCI compression at 75% bit rate	96
7.20	Reconstructed image of High operation point on only LSBS compression at 6% bit rate	98
7.21	Reconstructed image of High operation point on only LSBS compression at 75% bit rate	98
7.22	Reconstructed image of High operation point on only ECThread8 compression at 6% bit rate	100
7.23	Reconstructed image of High operation point on only ECThread8 compression at 75% bit rate	100

1 INTRODUCTION

The growing reliance on medical imaging in clinical practice has led to an increasing demand for efficient data management. As new technologies for image acquisition continue to emerge, they inherently generate the need for effective storage, processing, and transmission of the resulting data. Imaging modalities such as Magnetic Resonance Imaging (MRI) [1], Computerized Tomographies [2], and FIB-SEM Imaging [3] typically produce images with high resolution and bit depth, enabling the preservation of detailed and clinically relevant information for analysis and diagnosis. Consequently, this trend emphasizes the importance of developing reliable image compression methods that can reduce data size while maintaining the integrity of critical information.

However, due to their medical nature and the critical information they contain, such images require particular care. This includes the use of compression methods that preserve data integrity and do not compromise the content or interfere with sensitive tasks, such as organelle segmentation performed with the aid of Deep Learning (DL) models.

This Master's project aims to evaluate the impact of compression methods on medical imaging, alongside the application of a segmentation model trained for this purpose, in order to assess image degradation both from a human visual perspective and through the analysis of an Artificial Intelligence-based model. The primary objective is to examine the segmentation of the same images across different post-compression bit rates, identifying where information loss becomes most significant and determining whether image compression remains beneficial, ultimately contributing to the development of more efficient solutions for medical data management.

The data analysis will be performed through a mixed methodology. Qualitative methodology will visually sample the differences among the various steps done throughout the process, approaching the human vision to computer manipulation. A quantitative approach will be employed to extract values related to image segmentation and compression, and work with them to acquire better results that are not achievable through a qualitative analysis.

This document is structured in accordance with the steps taken during the project, as Figure 1.1 illustrates:

- The **State of the Art** chapter will present the state of the current technology, exploiting the three main elements of this work: FIB-SEM Imaging, Image Segmentation Methods, and Image Compression Methods;

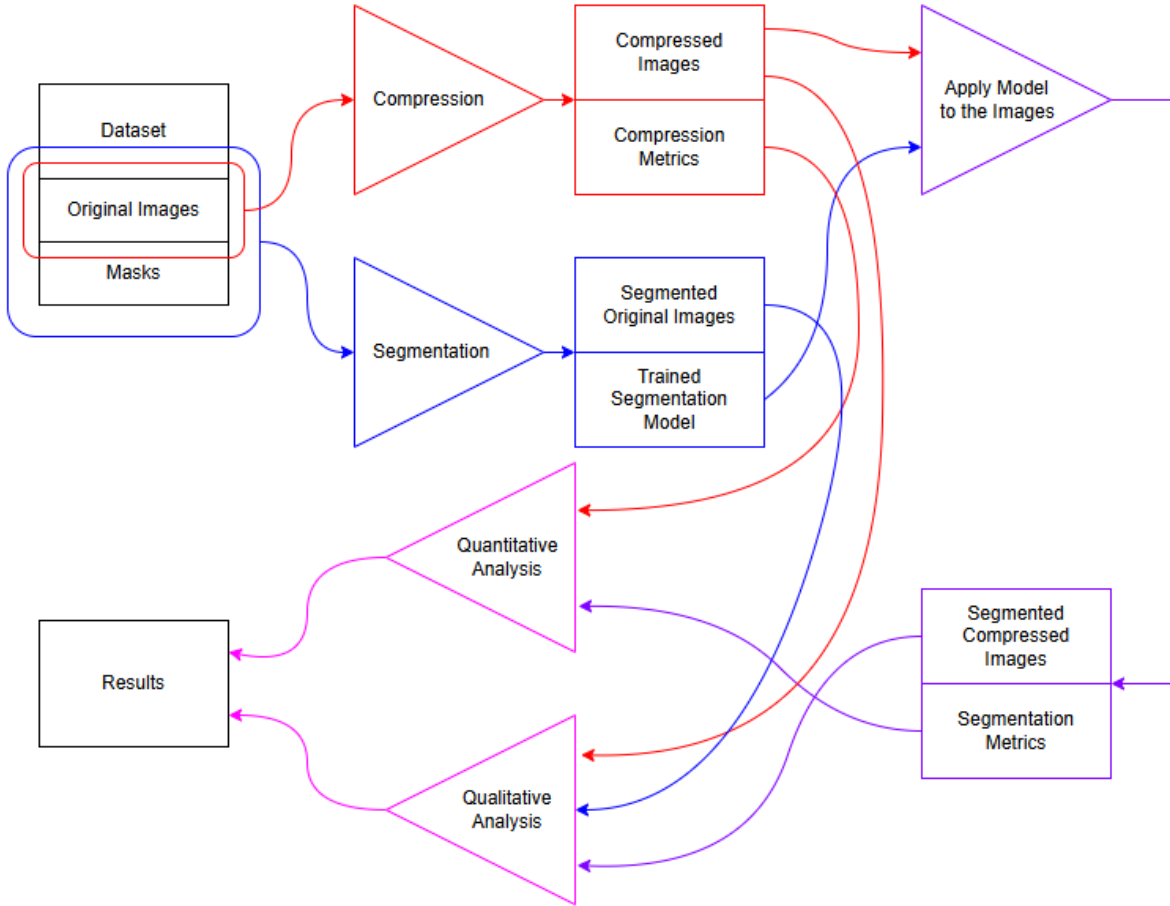


Figure 1.1: Summary of the elapsed procedure

- The **Dataset** section describes the origin of the acquired images, how they are structured, from file format to content, and the modifications applied to make them suitable for processing;
- The **Image Segmentation** section justifies the chosen segmentation model, describes the quality metrics associated to image segmentation and exposes the experiments done with the model;
- The **Image Compression with JPEG AI** chapter explains the compression code used, the quality metrics projected and its behaviour in the execution;
- The **Compressed Image Segmentation** chapter applies the model trained in **Image Segmentation**, applies it to the test images before and after compression and performs a quantitative analysis on the extrapolated metrics and a qualitative analysis on the segmented images, comparing the compression stages.

2 STATE OF THE ART

The images captured using FIB-SEM technology have gained increasing relevance in fields such as biology and microbiology, where this technique is widely used to study cells, intracellular organelles, and microorganisms, as highlighted in a review by C. Kizilyaprak et al. [4]. In the field of nanotechnology, Lirong Zhao et al. [5] emphasize the importance of FIB-SEM by enabling the precise assembly of components at the micrometer scale, addressing the growing demand for increasingly miniaturized electronics. Beyond these areas, FIB-SEM is also significant in materials science, where it supports the analysis of relationships between material properties, structure, processing, and performance, as discussed by M. W. Phaneuf [6]. Given its importance across multiple scientific domains, combined with the high quality and large volume of data produced, there is a clear need to develop robust segmentation and compression techniques that reduce storage requirements while preserving image interpretability.

2.1 FIB-SEM Imaging

FIB-SEM imaging relies on the use of a Focused Ion Beam (FIB) to remove successive layers from a sample surface, while a Scanning Electron Microscope (SEM) captures images of each newly exposed section, as depicted by Figure 2.1. This process is repeated along the depth of the sample, producing a sequence of images that can either be reconstructed into a 3D volume or analyzed individually. The work by Lirong Zhao et al. [5] also describes the composition and operation of both the FIB and SEM systems, including the types of ions used, the available operating modes, and the implementation of multi-beam configurations. These factors can significantly influence the outcome of the experiment, either by enabling different chemical interactions or by potentially damaging the sample.

The characteristics of the images produced by this technique are closely linked to the challenges they present. As reported by Graham Knott et al. [8], FIB-SEM images can achieve spatial resolutions below 10 nanometers per pixel, reflecting the technological advancements in the field. However, this high resolution often results in images with low contrast and significant noise, giving them an almost monotonic appearance that makes it difficult to distinguish between different structures. These images typically contain complex textures and irregular patterns, which increases the need for higher resolution and greater bit depth, ultimately leading to very large datasets, often reaching several gigabytes in size. As a consequence, automatic segmentation becomes

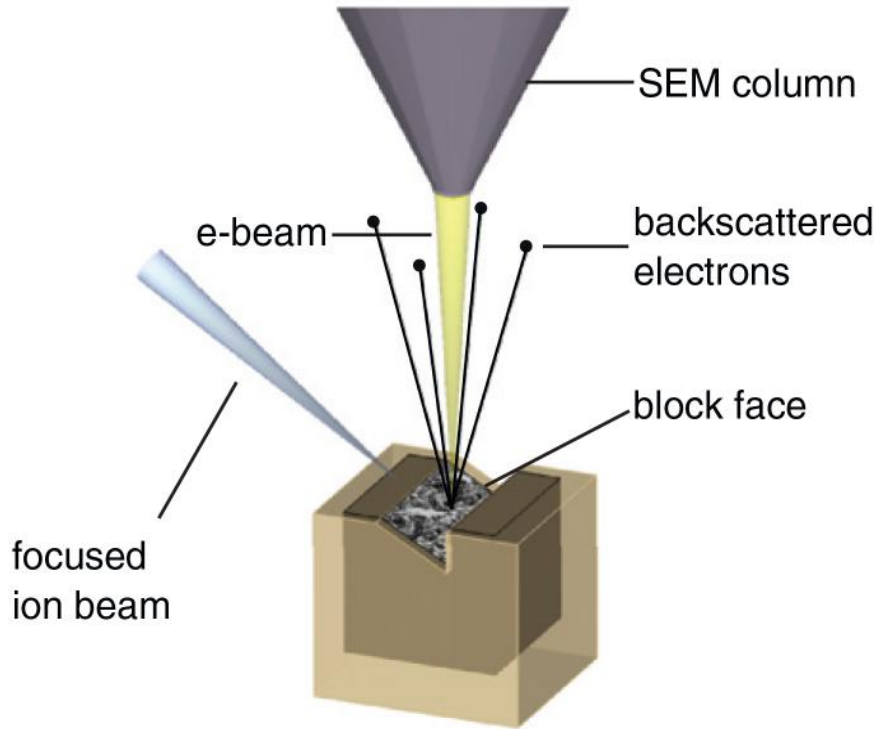


Figure 2.1: FIB-SEM Image extraction [7]

particularly challenging, as shadows and intensity variations hinder the performance of both traditional algorithms and neural networks. Figure 2.2 illustrates an example of a FIB-SEM image [9], where the low contrast, visible noise, and irregular structural patterns of organelles can be clearly observed.

2.2 Image Segmentation Methods

One of the main challenges associated with FIB-SEM imaging lies in the segmentation of the generated data. Image segmentation aims to partition an image into meaningful regions, typically corresponding to different structures or objects of interest. In the context of FIB-SEM data, this task is particularly complex due to the low contrast, high noise levels, and irregular textures present in the images. These characteristics make it difficult to clearly distinguish boundaries between structures, especially when dealing with biological samples where variations can be subtle and highly heterogeneous.

2.2.1 Classic Segmentation

Traditional image processing techniques, such as thresholding, edge detection, and region-based methods, often struggle to produce reliable results under these conditions. Their performance is typically limited by their reliance on intensity differences and predefined rules, which are not well suited to handle the variability and ambiguity found in FIB-SEM images. As a result, more advanced approaches have been explored

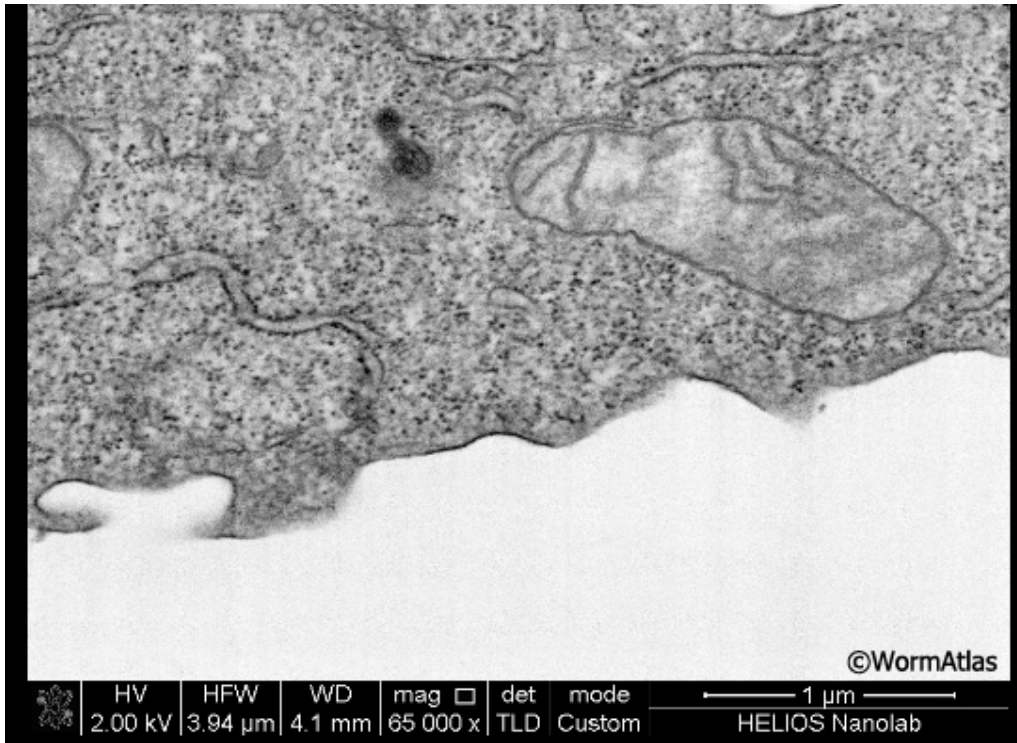


Figure 2.2: Example of a FIB-SEM image [9]

to address these limitations.

Thresholding is, as the name suggests, a technique that uses a threshold value to separate pixels into two groups, typically those below and those above the threshold. A study by Yue Liu et al. [10] reviews various thresholding algorithms, ranging from simple approaches such as manually selecting a threshold to more advanced methods like Otsu’s method, which evaluates all possible threshold values from 0 to the maximum bit depth minus one to determine the most suitable option. However, when applied to FIB-SEM segmentation, selecting an appropriate threshold becomes challenging. As illustrated in Figure 2.3, the histogram exhibits three distinct peaks at pixel values of 0, approximately 130–140, and 255. The peaks at 0 and 255 correspond to the black and white background regions at the top of the image.

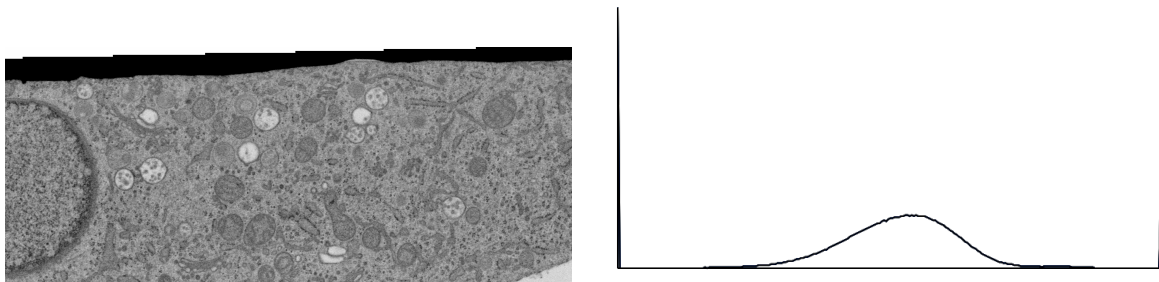


Figure 2.3: Histogram of a typical image from the dataset provided for this project

K-means clustering is an image segmentation algorithm used to identify K groups of

pixels within an image. By reshaping an RGB image of dimensions W by L into a $W \cdot L$ by 3 representation, clusters of pixels with similar characteristics can be identified. A value of K is then selected to define the number of clusters, with each cluster represented by a corresponding centroid.

Yao Zhang et al. [11] investigated several segmentation algorithms, including K-means clustering, using SEM images of different rock types. In their work, they selected $K = 2$, as the task involved segmenting a single class against the background. This effectively produced a thresholding-like result, where one cluster contained pixels below a certain intensity range and the other contained pixels above it.

In this report, K-means clustering was applied experimentally to the image shown in Figure 2.3.

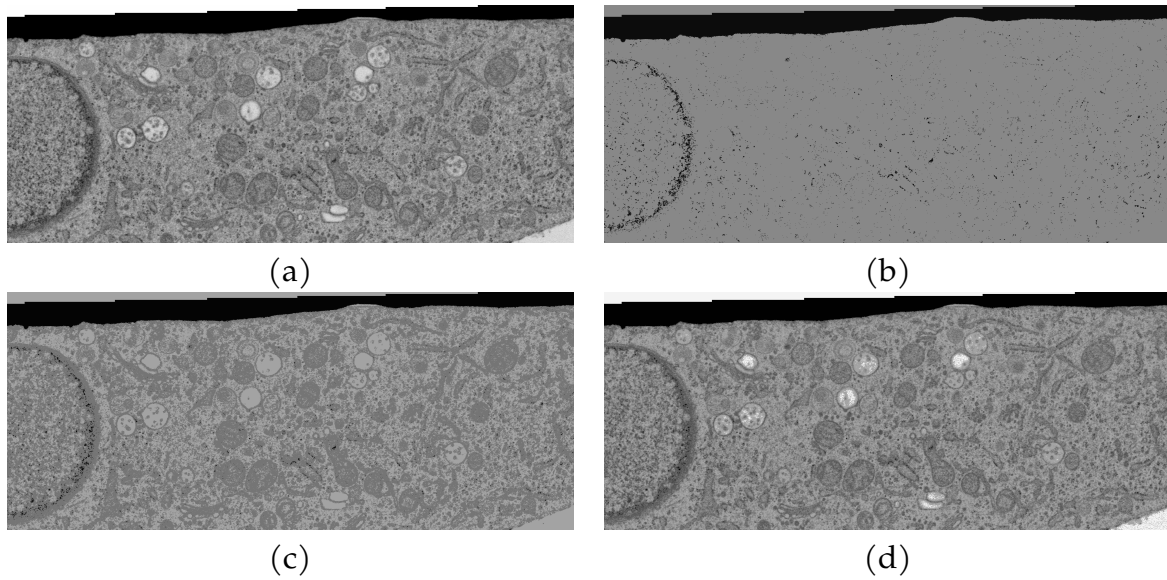


Figure 2.4: Applying K-means clustering to the image in Figure 2.3. (a) is the original image, (b) corresponds to $K = 2$, producing a segmentation that separates low-intensity pixels from high-intensity ones, in (c) the value of K was increased to 3 in an attempt to isolate the peaks at the extremes of the histogram, and in (d) K was further increased to 6 to evaluate whether the central region of the histogram could be divided into four distinct pixel clusters, the number of classes used in the project.

Figure 2.4 presents the results of applying the K-means algorithm to Figure 2.3. The results of this experiment indicate that K-means segmentation is not well-suited for FIB-SEM images, primarily due to their low contrast and high noise levels.

2.2.2 Deep-Learning Based Segmentation

As discussed in Section 2.2.1, classical segmentation algorithms are not well-suited for Scanning Electron Microscopy (SEM) image segmentation. This limitation arises from the irregular shapes of the structures of interest, the low contrast between the ground truth and the background, and the high levels of noise present in the images. These

challenges highlight the need for more robust and reliable approaches, leading to the widespread adoption of deep learning methods as the standard for SEM image segmentation.

FIB-SEM segmentation can be performed either by leveraging the full volumetric data and applying 3D segmentation techniques, such as 3D U-Net, or by dividing the volume into 2D slices and applying deep learning models like U-Net or YOLO to each slice individually.

U-Net, introduced by Olaf Ronneberger et al. [12] and represented in Figure 2.5, is a convolutional neural network designed for biomedical image segmentation. Its name originates from the U-shaped architecture, which is based on an encoder–decoder structure that enables effective semantic segmentation.

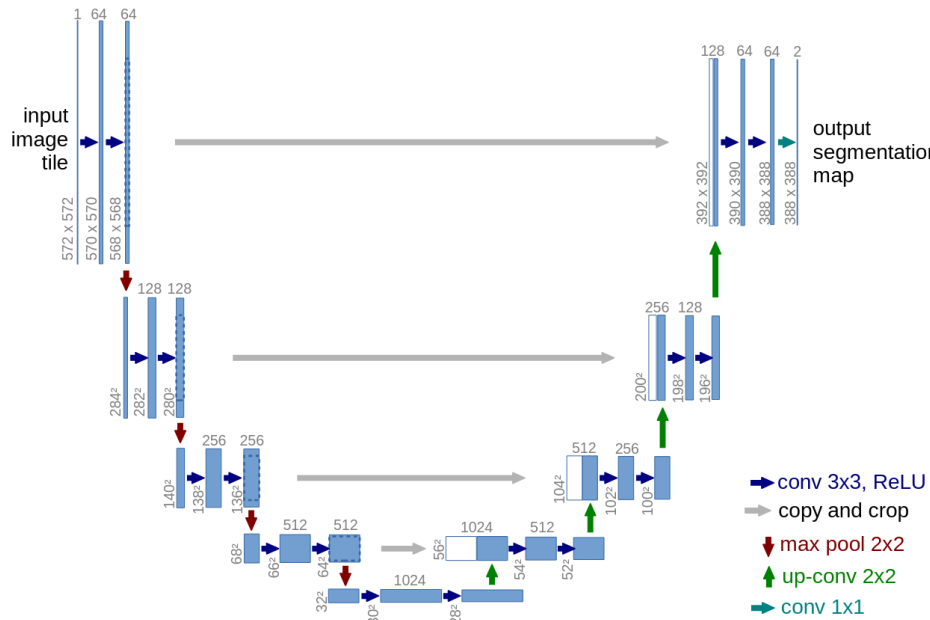


Figure 2.5: U-Net Architecture [12]

The encoder, located on the left side of the architecture, functions as a feature extractor by progressively reducing the spatial dimensions of the input image while increasing the number of channels, enabling the model to capture higher-level representations. The decoder, on the right side, reconstructs the original image resolution from the encoded representation, while integrating features from corresponding encoder layers to preserve spatial information.

A study by Qamar Yasin et al. [13] applied the U-Net architecture to FIB-SEM imaging, highlighting its reliability for segmentation tasks involving such data.

An extension of this approach, 3D U-Net, introduced by Çiçek et al. [14] and represented in Figure 2.6, demonstrated that volumetric convolutions improve consistency across slices, reduce topological errors, and capture contextual depth information not available in purely 2D models. By extending U-Net to three dimensions, 3D U-Net

performs convolutions on volumetric patches ($x \times y \times z$), enabling the model to learn inter-slice continuity in structural representations. This is particularly advantageous for segmenting organelles or membranes that span multiple slices, as it better preserves morphological coherence.

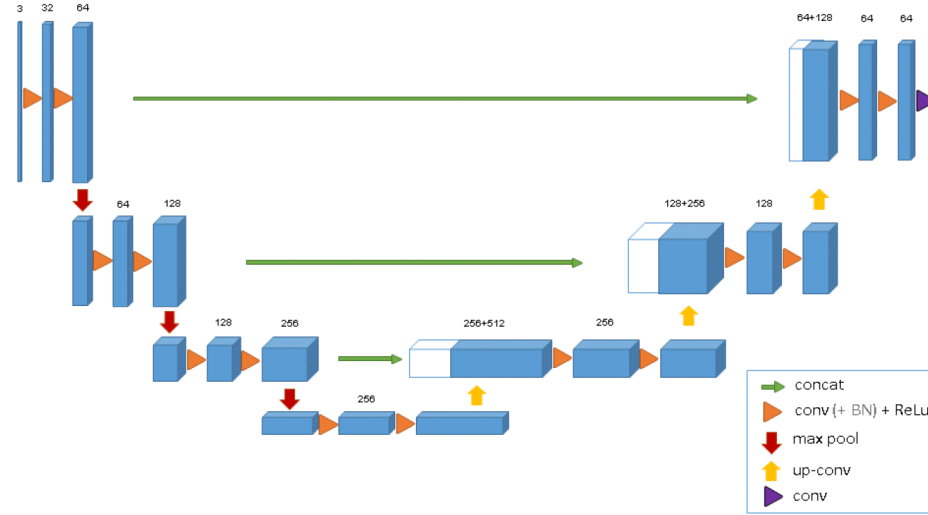


Figure 2.6: 3D U-Net Architecture [14]

A report by Jin Kousaka et al. [15] describes experiments involving the application of 3D U-Net to images generated using FIB-SEM technology. The authors developed a pipeline in which a 3D U-Net model segments six classes of cellular components from serial block-face FIB-SEM images, enabling the extraction and reconstruction of individual cells within large-scale volumes. The system successfully automated the generation of 3D models of cells and organelles, achieving high accuracy for larger structures, although challenges remained in the fine segmentation of smaller organelles and more complex division-stage images.

These models provide near-accurate semantic segmentation by grouping instances of the same class into a single segmented region. However, distinguishing individual elements within the same class—whether for counting, separation, or boundary definition—introduces an additional challenge. This is addressed through instance segmentation. The Ultralytics YOLOv8 model [16] offers a range of pre-trained models for instance segmentation and related tasks. Its user-friendly implementation, combined with precise boundary detection, makes it particularly effective for images with well-defined regions. Unlike U-Net-based architectures, YOLOv8 performs detection and segmentation jointly, making it inherently suitable for instance-level analysis.

When applied to FIB-SEM datasets, YOLO-style workflows enable efficient and flexible processing of large 2D or 3D image stacks, facilitating rapid localization of organelles, defects, and structural features. This capability supports region-of-interest extraction and downstream quantitative analysis at scales that would be impractical with manual

annotation alone.

In scientific imaging, YOLOv8 has seen increasing adoption. Ferji et al. [17] applied YOLOv8 with weighted box fusion to detect polymer nanostructures in TEM images, enabling high-throughput nanoparticle size estimation. Ruzaeva et al. [18] used YOLO for instance segmentation of dislocations in TEM micrographs, converting predicted masks into geometry-aware metrics for quantitative analysis. Similarly, Yu et al. [19] developed a semi-supervised YOLOv8 framework to detect micron-scale defects in industrial polymer films, demonstrating strong generalization with limited labeled data.

These studies highlight the potential of Ultralytics YOLOv8 to accelerate region-of-interest localization and instance segmentation in large and complex imaging datasets. In the context of FIB-SEM imaging, where massive 2D or volumetric datasets are common, YOLOv8's efficiency and flexibility make it a particularly promising tool for rapid detection and segmentation, while reducing the burden of manual annotation.

2.3 Image Compression Methods

Focused Ion Beam Scanning Electron Microscopy (FIB-SEM) produces extremely high-resolution volumetric datasets, typically acquired as sequences of 2D slices that together form 3D stacks. These datasets often reach sizes ranging from tens to hundreds of gigabytes, making efficient data compression essential for storage, transmission, and subsequent analysis—particularly when the images are used in automated tasks such as segmentation.

Image compression refers to the process of reducing the representation size of an image to facilitate storage and transfer. The two main types of compression are lossless and lossy methods. The scheme shown in Figure 2.7 illustrates the general workflow of image compression: the original image is processed by a codec, which includes both an encoder and a decoder. The encoder converts the image into a compressed binary representation, while the decoder reconstructs the image from this data. In the case of lossless compression, the reconstructed image is identical to the original, whereas lossy compression produces an approximation.

2.3.1 Lossless Image Compression Methods

Lossless image compression techniques are designed to reconstruct the original pixel values after decompression. A review by Baba Vemuri et al. [20] presents a range of such methods, including different linearization schemes that convert images into pixel sequences, as well as coding techniques such as Huffman coding, Lempel–Ziv–Welch (LZW), arithmetic coding, and run-length encoding.

Huffman coding begins by determining the frequency of each value in the dataset. A

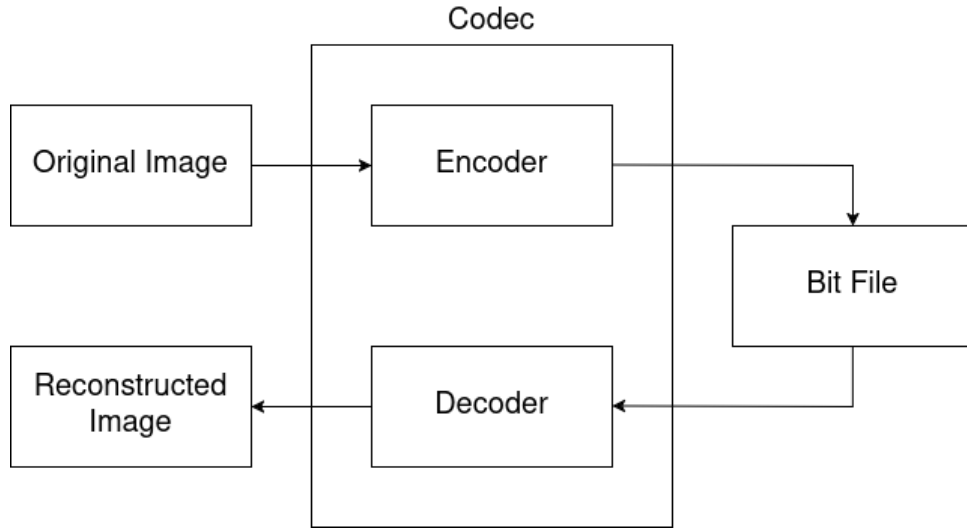


Figure 2.7: General workings of Image Compression

tree-like structure is then constructed, where values with lower frequencies are placed at the lower levels and grouped in pairs. The frequencies of these groups are iteratively combined until all values are incorporated into the structure. As a result, more frequent values are assigned shorter binary codes, while less frequent values receive longer codes, as illustrated in Figure 2.8. Each value is associated with a unique prefix, enabling accurate decoding and reconstruction of the original data.

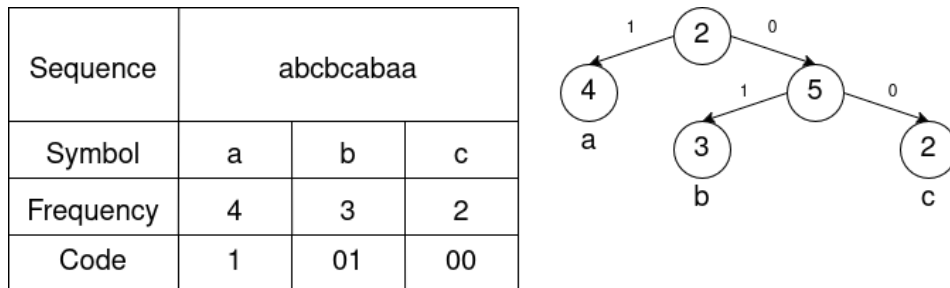


Figure 2.8: Huffman Coding

Lempel–Ziv–Welch (LZW) coding is a form of adaptive compression that dynamically assigns codes to symbol sequences as they are encountered. The method begins with an initial dictionary containing individual symbols, each mapped to a corresponding code. As the algorithm processes the data stream, it identifies recurring patterns and progressively adds longer symbol sequences to the dictionary. By the end of the process, the output consists of a sequence of codes representing both individual symbols and previously identified patterns.

Arithmetic coding, in contrast, represents an entire data stream as a single value within the interval between zero and one. Similar to Huffman coding, it assigns probabilities to symbols based on their frequency. These probabilities define subintervals within the main interval. As each symbol is processed, the interval is iteratively refined, resulting

in a progressively smaller range that uniquely represents the input data. The decoder reconstructs the original stream by reversing this process using the encoded value and the same probability model.

Run-length encoding (RLE) is a simpler technique that does not rely on symbol frequencies or probabilities. Instead, it compresses data by counting consecutive occurrences of the same symbol and encoding them as pairs of symbol and repetition count. For example, the string “lossless” would be encoded as “1l1o2s1l1e2s”.

2.3.2 Lossy Image Compression Methods

Lossy compression achieves significantly higher compression ratios by relaxing the requirement for perfect reconstruction. Annalakshmi N. [21] outlines three principal lossy compression techniques: scalar quantization, vector quantization, and transform coding.

Scalar quantization is the simplest approach, operating on individual signal samples such as pixel values or transform coefficients. The range of values is divided into intervals, and each sample is mapped to a representative value within its interval. This method is computationally efficient and widely used; however, it does not exploit correlations between neighboring samples, which can limit its performance on structured data.

Vector quantization extends this concept by processing groups of samples as vectors in a multidimensional space. Using a predefined codebook, each input vector is mapped to the closest representative vector. By capturing relationships between samples within a block, this method can achieve better compression efficiency for a given distortion level. However, it is more computationally demanding and requires substantial memory to store the codebook, which has restricted its use in large-scale standards.

Transform coding is one of the most widely adopted techniques in modern image compression. It applies linear transforms—such as the Discrete Cosine Transform (DCT) in JPEG or wavelet transforms in JPEG2000—to decorrelate pixel values. These transforms concentrate most of the signal energy into a small number of low-frequency coefficients, allowing more effective quantization. After transformation, coefficients are quantized, and less perceptually significant high-frequency components may be coarsely quantized or discarded. This approach reduces redundancy and enables high compression efficiency while maintaining acceptable perceptual quality.

Despite their effectiveness for human visual perception, these methods often struggle to preserve fine structural details at low bitrates. Artifacts such as blocking, ringing, and blurring can degrade high-frequency information, including edges, textures, and small anatomical structures in low-contrast images. Furthermore, their fixed transform pipelines limit their adaptability to machine vision tasks, often reducing the per-

formance of segmentation or detection models when images are heavily compressed. Traditional codecs also lack the flexibility to incorporate semantic or task-specific optimization, making them less suitable for workflows where compressed data must be interpreted by both humans and AI systems.

2.3.3 Learning-Based Image Compression

Learning-based image compression utilizes data-driven approaches, typically based on deep neural networks, to replace or enhance traditional hand-crafted compression pipelines. Rather than relying on fixed transforms, these methods learn compact latent representations directly from large training datasets.

Most contemporary approaches adopt autoencoder architectures, where an encoder network maps the input image into a lower-dimensional latent space. This representation is then quantized and entropy-coded, while a decoder reconstructs the image from the compressed data, as illustrated in Figure 2.9. By encoding the image into a compact form prior to compression, the amount of data to be processed is reduced, improving overall efficiency. Dibyendu Barman et al. [22] provide an overview of this approach by examining various applications of autoencoders.

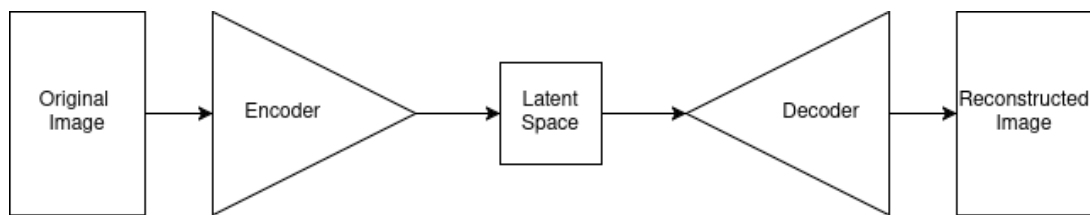


Figure 2.9: Generic representation of an Autoencoder

Modern neural image compression systems are built upon key architectural components that enable efficient, content-adaptive, and end-to-end optimized compression. At their core lies an autoencoder framework, in which an analysis transform (encoder) maps the input image into a compact latent representation, and a synthesis transform (decoder) reconstructs the image from this compressed form. Unlike traditional codecs that rely on fixed transforms, these neural transforms are learned, allowing them to capture complex spatial dependencies within the data.

A critical factor in achieving high compression performance is the use of hyperpriors and entropy modeling. The hyperprior network estimates statistical parameters, such as the scale or variance of latent variables, enabling more accurate probability models for entropy coding and improving compression efficiency. Zhengxue Cheng [23] discusses the role of hyperpriors in learned lossless compression. Additionally, many systems incorporate context models—such as autoregressive networks or masked convolutions—to further exploit spatial dependencies within the latent space. Together,

these components allow the codec to generate a bitstream that closely matches the true distribution of image features, thereby reducing redundancy.

Another important mechanism is learned quantization combined with rate–distortion optimization. Since quantization is inherently non-differentiable, as noted by Shangyu Chen [24], practical systems employ differentiable approximations during training to enable end-to-end optimization. Rate–distortion optimization provides a balance between bitrate and reconstruction quality, allowing fine control over compression levels for different applications. Michael Schäfer et al. [25] explore this concept by training autoencoder-based models that optimize both compression efficiency and reconstruction fidelity.

JPEG AI represents the most recent effort by the JPEG Committee to establish a next-generation, learning-based image compression standard. Proposed by João Ascenso et al. [26], this initiative reflects a broader shift toward data-driven methods capable of surpassing traditional codecs. Over the past decade, neural compression techniques—particularly those based on autoencoders with advanced entropy models—have demonstrated superior performance compared to classical standards such as JPEG and JPEG 2000, both in terms of bitrate efficiency and perceptual quality. For example, Zhengxue Cheng et al. [27] showed that an autoencoder-based model can outperform JPEG2000 on specific datasets, while Jooyoung Lee et al. [28] proposed an entropy model that improves Peak Signal-to-Noise Ratio and Multiscale Structural Similarity.

Despite these advances, the lack of standardization across neural compression methods has limited their widespread adoption. Differences in architectures, bitstream formats, and training procedures have hindered interoperability across platforms and applications. To address this issue, the Joint Photographic Experts Group (JPEG) launched the JPEG AI initiative, aiming to standardize machine-learning-based image compression. A Call for Proposals (CfP) [29] invited contributions from academia and industry, with candidate models evaluated on standardized datasets, objective metrics, and computational constraints. The strong performance of these approaches in both human-perceptual and machine-oriented tasks confirmed the potential of neural compression as a standardized solution.

An overview of the JPEG AI framework is provided by Semih Esenlik et al. [30], including the architecture illustrated in Figure 2.10, which details the internal structure of the standard.

The encoder initially separates the input image into its luminance component, x_Y , and chrominance component, x_{UV} . Analysis transforms are then applied to produce latent representations for luminance, y_Y , and color, y_{UV} , facilitating more efficient compression. These latent representations are passed to hyper-encoders, which generate tensors describing their statistical distributions.

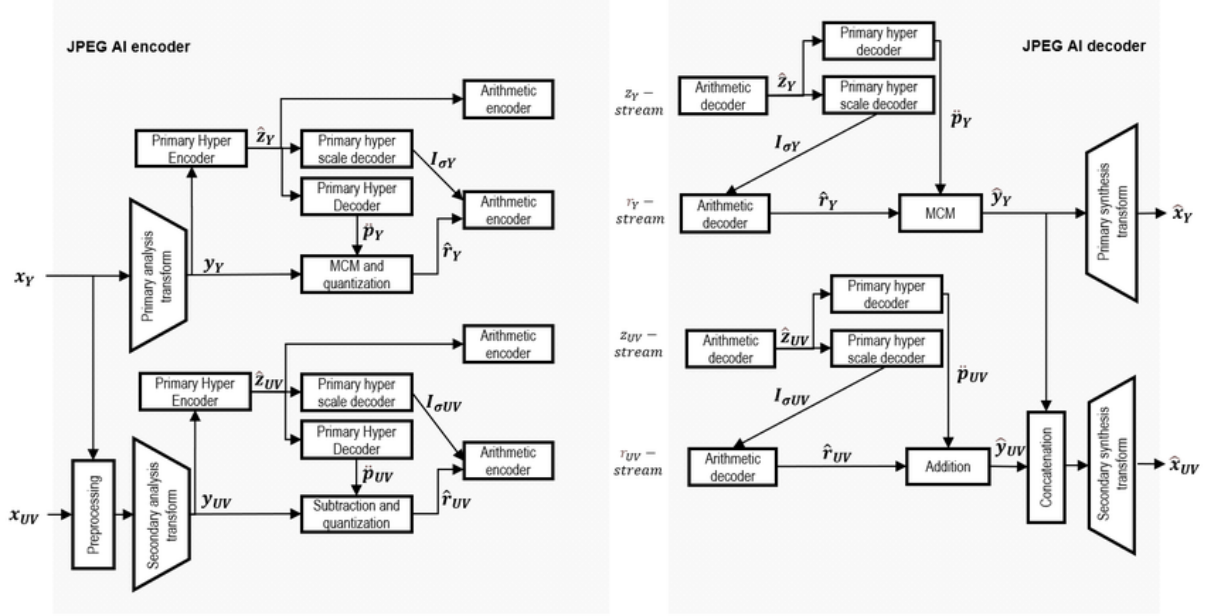


Figure 2.10: JPEG AI Framework [30]

These tensors are utilized in three ways: they are sent to an arithmetic encoder to generate bitstreams for the corresponding hyper-decoders in the decoder stage; they are processed by a primary hyper-decoder; and they are used by a primary hyper-scale decoder to estimate distribution parameters, namely $\hat{p}_Y$, $\hat{p}_{UV}$, $I_{\sigma Y}$, and $I_{\sigma UV}$. These parameters guide the quantization process and support subsequent encoding stages. Using this information, the latent representations are quantized and further encoded into compressed bitstreams. The decoder then performs the inverse operations in reverse order to reconstruct the original image.

JPEG AI offers several advantages for FIB-SEM data by leveraging learned representations that adapt to the statistical properties of electron microscopy images. Its learned transforms enable better preservation of fine textures and intensity variations, while advanced entropy models improve the handling of structured noise, reducing the likelihood of over-compression of important microstructural details. Furthermore, JPEG AI supports task-aware optimization, making it well-suited for workflows where compressed images are directly used by segmentation models. By jointly optimizing rate-distortion and task performance, it can maintain segmentation accuracy even at high compression levels, outperforming traditional codecs that often introduce artifacts detrimental to boundary detection.

3 DATASET

The initially obtained dataset [31] was comprised of seven TIFF files referring to a volume extracted from a HeLa [32] cell utilizing FIB-SEM [3] technology. Figure 3.1 demonstrates three images acquired through FIB-SEM technology, with Figure 3.1.(c) being one of the sample images with the dataset, and the remaining ones created by analyzing the volume across the remaining planes, extrapolating the other perspectives by reading the pixels in the height and width of the original.

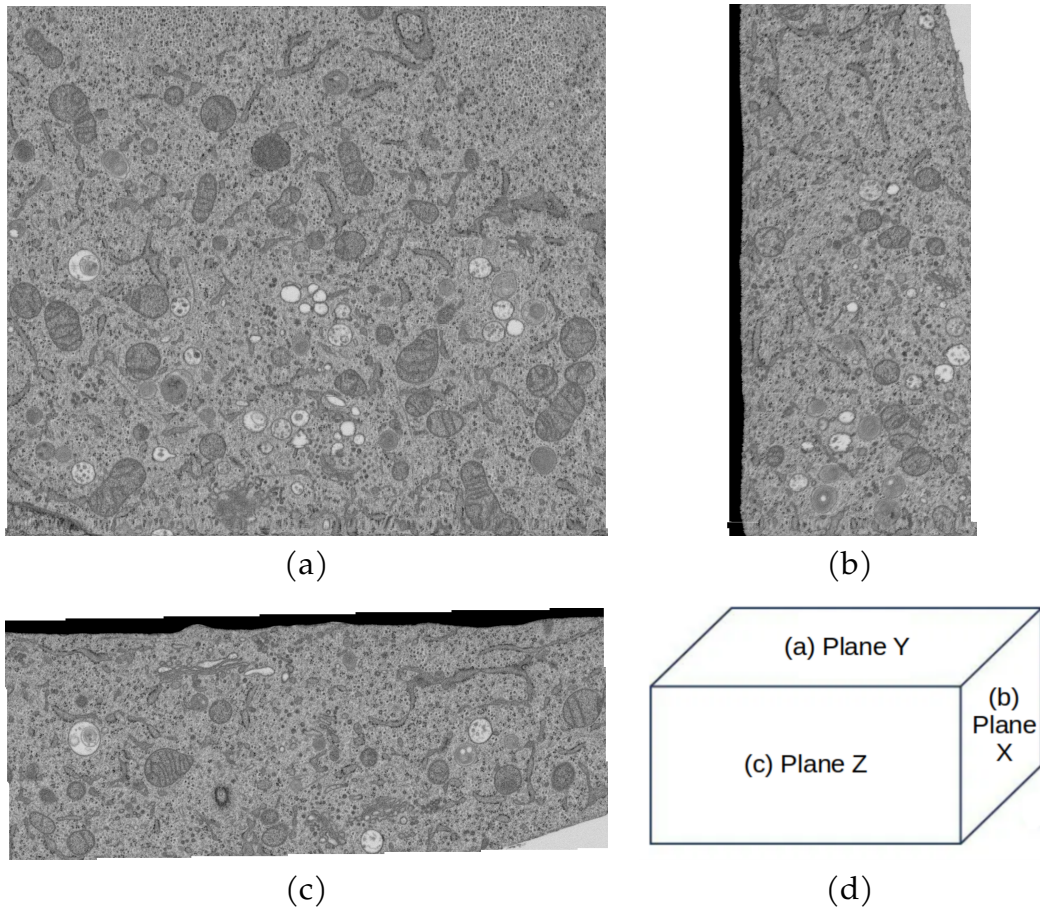


Figure 3.1: Three planes of the volume intercepted by the center point: (a) is extracted from a top perspective, referred as Plane Y; (b) is extracted from a side perspective, referred as Plane X; (c) is extracted from a front facing perspective, equating to the original set and referred as Plane Z; and (d) represents how the perspectives were obtained from the volume.

Images obtained through FIB-SEM, or Focused Ion Beam Scanning Electron Microscopy, technology are segments of a volume extracted using an electron microscope to sweep a surface to generate an image of the analyzed area. Consequently, a focused ion beam

removes the evaluated surface, allowing a new section to be captured with the microscope [33]. This technology emerged as a solution to the need for creating sample volumes, which was previously impossible with only an electron microscope. Due to its high precision and advanced planning before using it, this approach requires more time to obtain the volume, resulting in a high-quality, high-resolution transversally segmented solid capable of originating a three-dimensional volume for a more ascertained study [34].

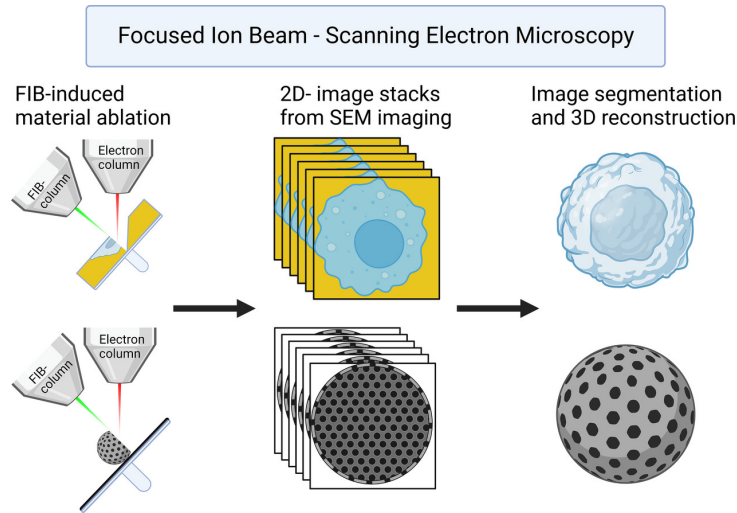


Figure 3.2: Image extraction model with resource to FIB-SEM technology [33]

The seven files are comprised of one file with the original volume, containing all the information of the volume present in the file, as illustrated by Figure 3.3, and the remaining files contain the masks of the most important organelles: the Mitochondrion, the Lysosome, the Early-endosome, the Late-endosome and the nucleus, with five files presenting the information individually for each organelle and one file grouping the files into one differentiating the information through pixel values. The masks present in the six final files are displayed as binary images to highlight the information to capture in the original file.

The images were extracted to usable files with a PNG extension utilizing the tool ImageJ [35], an open-source software created for easy image manipulation, allowing for interaction from 8 bits per pixel to 32 bits per pixel images, being able to read information from an array of extensions like TIFF and having the capability to transform each image in the TIFF file into a PNG file.

These extracted images have a width of 2023 pixels and a height of 846 pixels, with a depth of 16 bits per pixel across one channel. The segmentation model and the compression tool chosen for this experiment can only work with a bit depth of 8 bits per pixel, effectively reducing 65536 intensity tones to 256 intensity tones. The adopted solution for the problem was to operate with the eight most significant bits of the pixel. A study conducted by Lu Zenghai et al. [36] states that reducing the bit depth on an

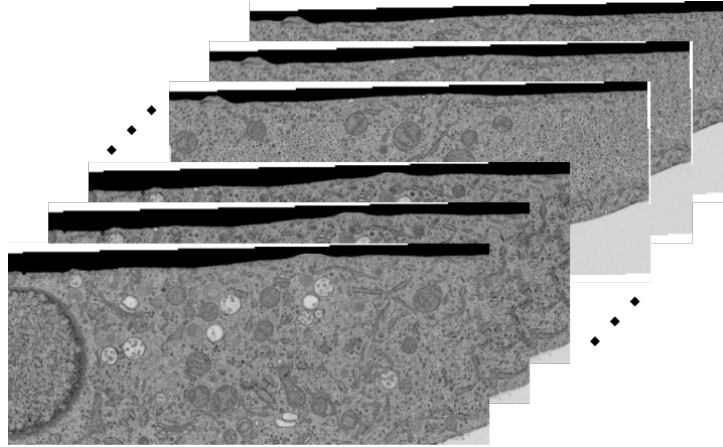


Figure 3.3: Appearance of the TIFF file with the images captured by FIB-SEM technology

image may cause the loss of some details in low contrast images, which, by observing Figure 3.3, can be concluded to be the case. The study compared 14 bit per pixel images to 8 bit per pixel images and observed differences in the Mean Absolute Error, a measure that evaluates the mean modal difference between the predicted value and the real value [37], and in the Mean Quadratic Error, an average of the square of the differences between the predicted value and the real value [37], indicating degradation in the quality of the image post bit depth reduction. However, the study also concluded that the error between the metrics calculated for the 14 bit per pixel images and the 8 bit per pixel images was negligible, for which they noted that working with 8 bits per pixel would not make a considerable difference. Although the initial bit depth of 16 bpp is higher than the 14 bpp maximum used in the study, this choice of reducing the working bit depth to 8 bpp was made because the potential loss of information was considered negligible. The segmentation masks already indicate the locations of the organelles, and for compression purposes, having both the original and compressed datasets is sufficient to obtain the desired results. In Figure 3.4, it is possible to see that visually the difference is minimal to the human eye.

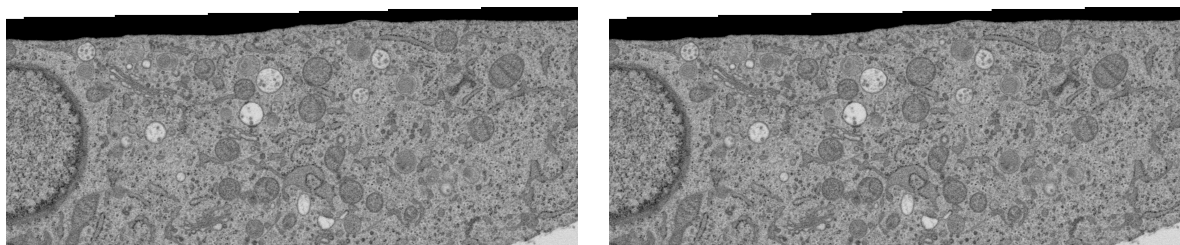


Figure 3.4: Visual difference between 16 bit per pixel images (left) and 8 bit per pixel images (right)

The organelles present in the images are accounted as described in Figure 3.5, with the bars reflecting the numerical amount of each organelle, exposed on the left axis, and

the orange line adding complement with a normalised representation.

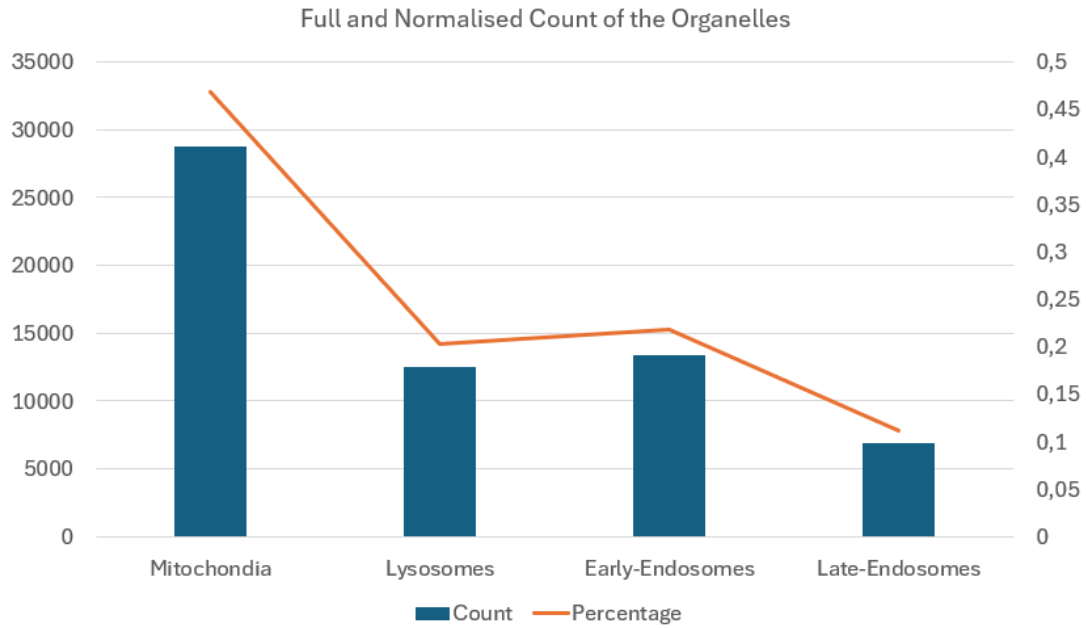


Figure 3.5: Numerical variance among the four studied classes

Figure 3.6 highlights with more detail the visual differences of the organelles when centered in 512 by 512 pixel images.

As can be observed from Figure 3.1, the nucleus appears sliced in the bottom left corner of (a), which is visible in all the images of the referring mask. This is because the volume was acquired with the nucleus located on an edge of the volume, making it impossible to get a full shot of the nucleus without deploying operations that would most likely be harmful to the content of the image, like analyzing the nucleus from an angle that captures full sections. Because of this, it was decided not to work with the nucleus and only train the segmentation model on full organelles.

According to a study conducted by Xinsheng Wang et al. [38], the Mitochondrion is an organelle essential to the cell. Named as "The powerhouse of the cell", the Mitochondrion is responsible for supplying energy to the cell through adenosine triphosphate, regulates cell death, transfers calcium to the cell, and segmentation of the Mitochondria allows for counting and qualifying their number, volume, and shape to assess the state of the cell.

The Lysosomes are organelles distributed in the cytoplasm of the cell that degrade residues present in it and act like the cell's digestive system. A study made by Qinle Ba et al. [39] aims to study the behavior of Lysosomes to visualize how they perform their tasks. Like the Mitochondria, their segmentation helps to qualify them, checking if the majority are grouped or scattered, as the study aims to discover.

The Early endosomes, also known as the classification endosomes, are responsible for

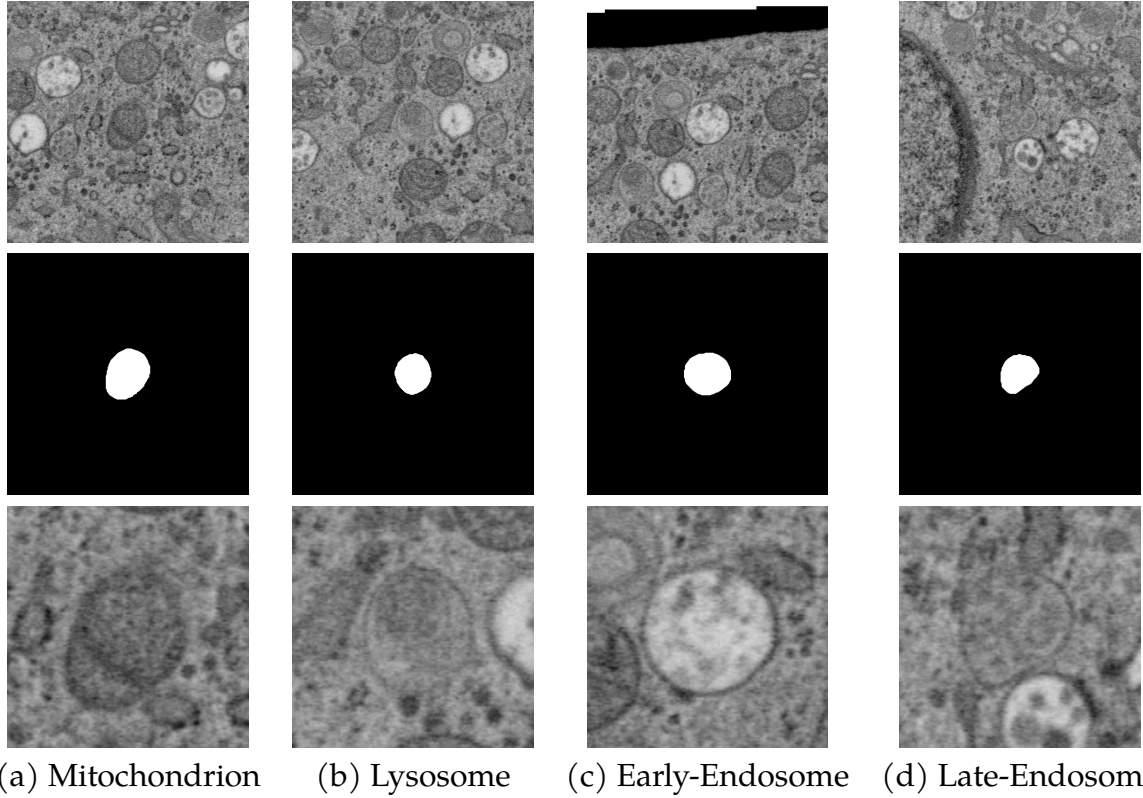


Figure 3.6: Visual contrast among the organelles and their respective location as indicated by the corresponding mask. (a) is the Mitochondrion, the most abundant organelle of the dataset, presenting high contrast with the background, allowing for easier detection; (b) is the Lysosome which hides with ease in the background, requiring extra care with its handling; (c) is the Early-Endosome and is usually clearer than the background, allowing better identification; and (d) is the Late-Endosome which is similar to the Lysosome in which it also blends with the remainder of the image thus needing more care.

deciding the destination of internalized proteins and lipids, sending them to the Lysosomes for disposal, recycling for the cell, or transferred to other cells, as indicated by a study from Marko Jovic et al. [40]. Their segmentation can aid in the study of the load dynamics, as well as determine if it is being discarded, recycled or stored.

Late endosomes are Early endosomes that reached a more acidic pH and thus matured, as stated in an article published by Cameron C. Scott et al. [41]. Their main task is to send the lipids and the proteins to the Lysosomes for disposal. Their segmentation evaluates their behavior and prevents irregularities that may harm the integrity of the cell.

4 IMAGE SEGMENTATION

Image segmentation plays a central role in the analysis of biomedical images and is essential for extracting meaningful structures from complex datasets such as those produced by Focused Ion Beam–Scanning Electron Microscopy (FIB-SEM). Given the high level of detail and the volume of data in FIB-SEM acquisitions, an effective segmentation approach must balance accuracy, computational efficiency, and practical usability. For this reason, in this work the Ultralytics’ YOLOv8 framework for segmentation was chosen.

YOLOv8 offers several advantages that align well with the goals of this project. It provides a unified pipeline capable of performing both object detection and instance segmentation, allowing detection-based workflows to naturally extend into segmentation tasks. The framework is also well-supported, with extensive documentation, an active user community, and many tutorials that facilitate rapid experimentation and troubleshooting. Furthermore, its modular design and straightforward training procedures make it suitable for iterative experimentation, which was particularly important in this study as different strategies were tested to improve model performance on a specialized biomedical dataset.

This chapter is structured into three main parts, each corresponding to a different stage of the segmentation workflow explored during the project:

- **YOLO Detection and Fundamentals** introduces the principles of object detection and covers the essential concepts behind the YOLO family of models. Here, the initial experiment carried out to understand the behaviour of the detection network and establish a baseline for later segmentation tasks is described. The work described in this section was presented in the 29th edition of the Portuguese Conference on Pattern Recognition (RecPad 2023).
- **YOLO Segmentation with Data Augmentation and Multi-Plane Sampling** focuses on the segmentation experiments, where YOLOv8 was trained using custom data augmentation techniques. The available training data was expanded by generating additional images from the FIB-SEM dataset through re-slicing in different orthogonal planes, enabling the model to learn from a broader set of structural variations. The work described in this section was presented in the 30th edition of the Portuguese Conference on Pattern Recognition (RecPad 2024).
- **YOLO Hyperparameter Tuning** examines the impact of hyperparameter optimization on segmentation performance. By systematically adjusting key training

parameters, model behaviour changes and configurations that improved segmentation quality were evaluated.

4.1 YOLO Detection and Fundamentals

Before exploring image segmentation with YOLOv8, a basic comprehension into it's default behaviour was needed to assess how to adapt the data, alter the arguments and interpret the results. The pre-trained models of the YOLOv8 family are trained on Common Objects in Context (COCO) images, which are detailed by Tsung-Yi Lin et al. [42] and consist of images of objects found in day-to-day life. These images are selected to aid in creating classification, detection, semantic and instance segmentation models like the ones used by the YOLO model family. The region boundaries are well defined and the images contain high contrast, facilitating the task for what they were chosen and ensuring a good trained model with minimal input from the trainer. This is because the training images are coupled with precise locations of where the regions are in the image in the form of extra files added to the training dataset, the validation dataset and the test dataset. The training parameters play an important roll on the development of the training by ensuring YOLO resizes the images to the provided new size, trains with more images at once thus loading the model with more information about the dataset, controlling the duration of training and how fast the model trains. Once trained, the validation set will produce evaluation metrics to analyse the model's training, offering insight into classification, loss, and an understanding into the relation among the true ROI and the predicted ROI.

Preparation of the Dataset

The YOLOv8 dataset is separated into three main sets, as illustrated by Figure 4.1.

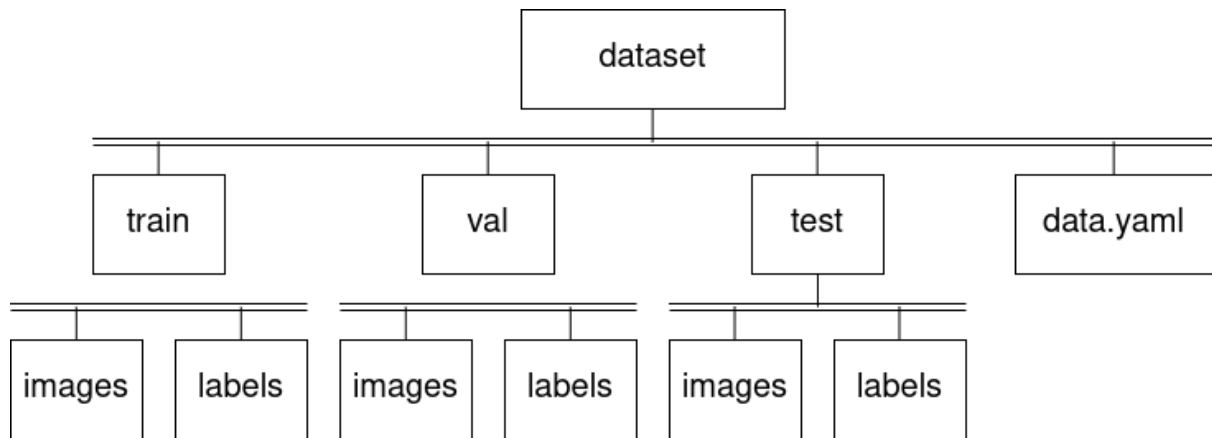


Figure 4.1: Representation of the file tree of the YOLOv8 Dataset

The "train" set includes the images the model uses during learning. Through repeated exposure, the model adjusts its internal weights and biases to enhance performance

and minimize errors. Because of this it must contain considerably more data than the other sets to study while allocating sizable validating and testing content, so a balance must exist among the training set and the others. The "val" set corresponds to the validation set where, between training epochs, the model will be tested on how the training proceeded and which feedback will influence the model's parameter adjustments during training. The "test" set will contain the remaining images which will be used to test the model once the training ended, being used to manually measure the models performance. Each of these sets contain two folders. The "images" folder holds the images to be accessed by YOLO during the training, validation and test, and they are required to contain at least one instance of the region to be detected. The "labels" folder possesses the text files with the information regarding the ROI in the images. The path to these folders are displayed in the file "data.yaml" along with information on the amount and name of the classes. This file will relay to the model the images for the training and the validation, as indicating the test set on the "data.yaml" file is optional.

The dataset used was divided into a training set consisting of 1066 images, a validation set of 356 images and a test set of 356 images, corresponding respectively to 60%, 20% and 20%. This ensures a majority of the data will be used to train the model, an amount equivalent to 30% of the training set size will be used to validate it between training stages and post training, and the same amount will be used to manually infer on the model's success rate.

The labels files adhere to a specific format in YOLOv8 detection. The information regarding each ROI must be written according to the structure "[class] [x] [y] [width] [height]". The value "class" represents the ID value attributed to the type of ROI, as in multi-class training each region must be represented as a class with an identifiable value. For this project the class for Mitochondrion was given the ID 0, Lysosome was given ID 1, Early-Endosome was given ID 2, Late-Endosome was given ID 3 and the Nucleus was given ID 4. The values "x" and "y" represent the top left pixel of the bounding box of the ROI, where "x" points to the coordinate along the abscissa and "y" to the coordinate along the ordinate with the origin in the top left pixel of the image. "Width" and "height" correspond to the width and height of the ROI's bounding box, usually being determined as the absolute difference between the pixel closest and the pixel furthest from the origin along the indicated axis. The values "x" and "width" are normalised in accordance to the width of the corresponding image while "y" and "height" are normalised by the height before being introduced into the text file. This change is due to YOLO performing alterations to the images like resizing and, if instructed, geometrical transformations and colour alterations.

To obtain the text files for the images a Python code was employed with Open-CV [43] to perform pixel-wise operations on the images and obtain the required values for the file's content. Since at the time the author was inexperienced with Python coding and

handling of multiple pixel regions independently of one another this implementation was reserved only for the nucleus, as in every image it appears there is only one instance of it. The remaining masks were handled by the software Sherlock [44]. Sherlock is a machine vision software capable of recognizing symbols in images like letters and numbers, read bar codes, allowing for identification of abnormalities in images and, pertinent for this project, visual control on the Regions of Interest being detected, values being extracted and handling and saving of values in a text file with the aid of Javascript code.

Figure 7.1 present in Annex A shows an instance of an image relating to a Mitochondrion mask being analysed by Sherlock. To obtain the values needed for the model, the software must be made aware of where the Regions of Interest reside, offering ROI tools so the user can visually select where to extrapolate content from. To ensure every ROI is detected regardless of position in the image, a rectangle ROI was placed covering the entirety of the images. The rectangle disposes of preprocessors and algorithms to aid in the task. Since the masks are 8 bpp binary images a Threshold preprocessor was activated to turn the image into regions Sherlock could interpret. To obtain the values a "Connectivity Binary" algorithm was employed which searches for white regions in the image and returns width, height, center point, count and other numerals relative to each blob. Those values can be extracted into variables for easier managing and usage in Javascript code. The code is responsible for taking in the variables set by the user to be extracted from the region and performing the needed calculations and variable type transformations to write them in text files associated with the image.

Argument Settings and Training

The training method available in Ultralytics [45] utilizes input arguments to make the training section more dynamic or less dynamic while also allowing for user customization. Some arguments serve to alter and influence aspects in the model and others to help the model interpret the provided images, with the main goal of obtaining the best conditions to train the model.

There are five variants of YOLOv8 models designed for detection tasks, denoted as n (nano), s (small), m (medium), l (large), and x (extra-large). These variants differ primarily in model capacity and training configuration, which in turn affects their performance across several key metrics. Table 4.1, acquired from a tutorial by Shaoni Mukherjee summarising the YOLOv8 model family [46], presents these metrics and highlights the distinctions among the models.

In this table, *Size* represents the input image resolution used during training, which is kept constant to allow a fair comparison under similar training conditions. The *mAP* metric, examined in detail later in this document, quantifies the overall detection accuracy of each model. *Speed* reports the inference performance when executed using

Table 4.1: Difference among the detection models of YOLOv8 [46] for the COCO set

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	18.4	142.4	1.21	3.5	10.5
YOLOv8s	640	27.7	183.1	1.40	11.4	29.7
YOLOv8m	640	33.6	408.5	2.26	26.2	80.6
YOLOv8l	640	34.9	596.9	2.43	44.1	167.4
YOLOv8x	640	36.3	860.6	3.56	68.7	260.6

CPU ONNX and A100 TensorRT runtimes. *Params* denotes the number of trainable parameters in the network (in millions), which directly correlates with the model’s computational complexity. *FLOPs* (floating-point operations per second), expressed in billions, indicate the number of computations required to process a single image.

Overall, the table illustrates a clear trade-off within the YOLOv8 family, where models designed for speed and reduced computational demands contain fewer parameters and consequently achieve lower accuracy, whereas larger models with more parameters provide improved performance at the cost of slower inference and higher resource requirements.

Among the available YOLOv8 variants, the YOLOv8n model (loaded from yolov8n.pt) was chosen. This is the smallest and most lightweight version of the YOLOv8 family, containing fewer parameters and reduced computational complexity when compared to the small, medium, or large variants. The use of YOLOv8n allows faster experimentation and lowers memory consumption while still preserving the core architectural improvements of YOLOv8, such as an anchor-free detection head and an optimized backbone-neck design.

The batch size, *batch*, was configured using YOLOv8’s automatic batch size selection (auto-batch). Batch size determines the number of images parallel processing during each forward and backward propagation. Larger batch sizes can improve training stability by providing smoother gradient estimates, but they also increase GPU memory requirements. Conversely, smaller batch sizes reduce memory usage but may introduce higher gradient variance, potentially slowing convergence. By enabling auto-batch selection, the framework dynamically determines the largest feasible batch size based on the available hardware resources. This approach ensures efficient utilization of memory without manual tuning, and reduces the risk of Out-Of-Memory errors, which is especially relevant when training with high-resolution medical images.

The model was trained for 100 *epochs*, where one epoch corresponds to a full pass through the entire training dataset. The number of epochs controls how long the model is allowed to learn from the data. Too few epochs may result in underfitting, where the model fails to capture relevant features, while too many epochs can lead to overfitting, where the model performs well on training data and underperforms on test data, particularly in datasets with limited variability. A value of 100 epochs, commonly used as the default value, represents a commonly adopted compromise in object detection tasks, providing sufficient opportunity for convergence while maintaining a reasonable training time. This duration allows the network to progressively refine both low-level and high-level feature representations without excessive risk of memorizing the training data.

The input image size, *imgsz*, was set to 2023×846 , matching the native resolution of the dataset images. Image size plays a critical role in object detection performance, as it directly affects the level of spatial detail available to the model. Higher resolutions preserve fine-grained structures and small objects, which is particularly important in low contrast images due to the faintness of the boundaries. However, increasing image size also leads to higher computational cost and memory usage. By retaining the original resolution instead of resizing to a standard square input, this configuration avoids unnecessary interpolation artifacts and ensures that anatomical proportions and spatial relationships are preserved during training.

All remaining training parameters were left at their default YOLOv8 values. Using these defaults minimizes the introduction of variables and ensures that performance differences observed in the experiments can be more confidently attributed to the factors under study rather than to aggressive or task-specific hyperparameter tuning. This design choice enhances the methodological robustness and reproducibility of the experimental results.

Results

After the training and validation processes, the model presented an array of metrics such as a Confusion Matrix to better understand what the model classified and misclassified. Figure 4.2 was obtained from this experiment. The organelle best predicted by the model was the Nuclei, mostly because of the 1778 images provided in the dataset, 117 contained the Nuclei and its size is considerably greater than that of other organelles, which allowed the model to accurately detect them and avoid confusion with other classes. The problem with this approach arose with the Background and the Mitochondria, where 51% of the background instances was classified as Mitochondria. Additionally, 28% was classified as Lysosomes, 16% as Early-Endosomes, 6% as Late-Endosomes, and 0% was declared as the true element. One explanation for these values comes from image noise. The most common noise in FIB-SEM imaging is Gaussian

Noise [47]. Gaussian Noise follows a Probability Density Function, affecting every pixel of the image. These changes in value favour merge pixel regions, affecting the model's perception of what it must detect, in this case masking the Background as Mitochondria or other organelles and preventing the model from correctly identifying it.

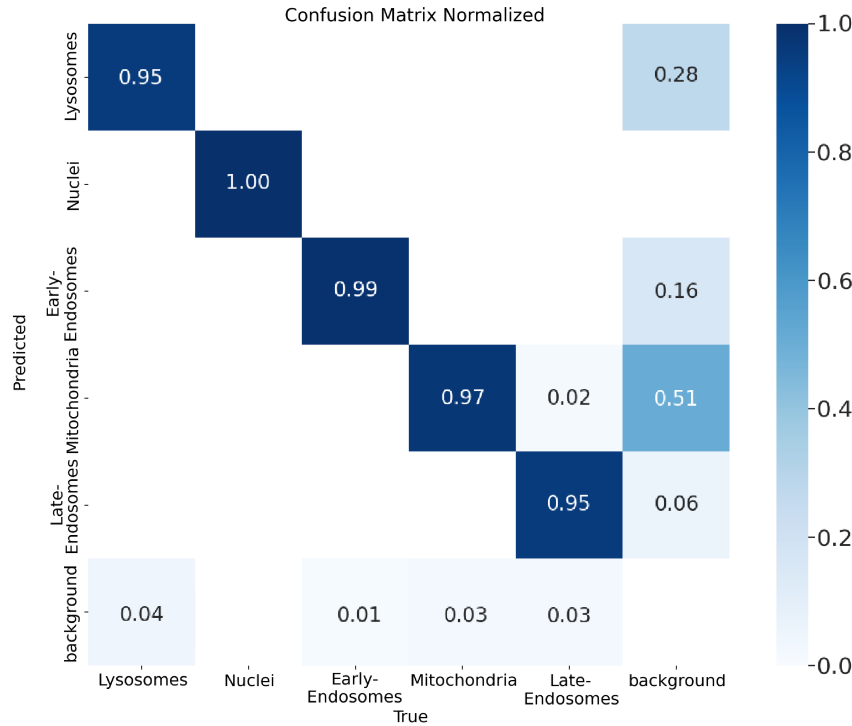


Figure 4.2: Normalised Confusion Matrix of Validation of Detection Task

Figure 4.3, credited to the work by Da Hoon Seol et al. [48], illustrates the distinction between TP (True Positives), FN (False Negatives), FP (False Positives), and TN (True Negatives), representing pixel classifications based on the comparison between the model's predictions and the ground-truth labels. Additionally, it presents some of the formulas for the evaluation metrics used to assess the model's performance. Precision measures the proportion of correctly predicted positive instances among all predicted positives, indicating the model's predictive reliability. Recall measures the proportion of correctly predicted positive instances among all actual positives, reflecting the model's ability to identify relevant pixels. Accuracy represents the proportion of correctly classified instances over the total number of instances, while the F1 Score corresponds to the harmonic mean of Precision and Recall, ensuring that both metrics must be high for the model to be considered effective.

Figure 4.4 presents the F1-Confidence curve, a plot illustrating the relation between the F1 Score, whose formula is present in Figure 4.3, and the different Confidence Thresholds. This curve is interpreted by analyzing what values of confidence achieve a higher F1 Score and what value maximizes it. The model can then be set to only classify a

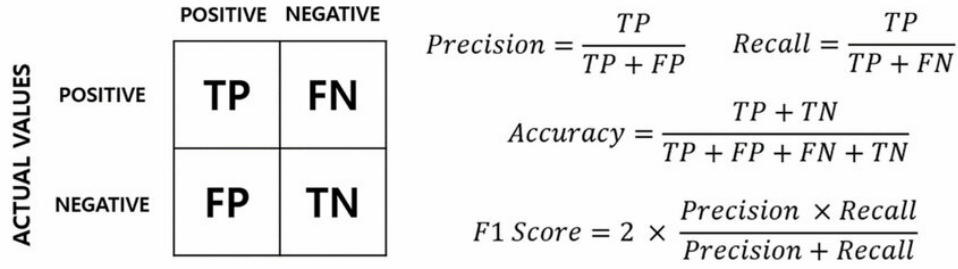


Figure 4.3: Interpretation of a Confusion Matrix and usage of presented values for Precision, Recall, Accuracy and F1-Score

region if its confidence in the classification is higher than that confidence threshold. Figure 4.4 also indicates that a Confidence value of 40% to 80% maximizes the F1 Score according to the "all classes" line.

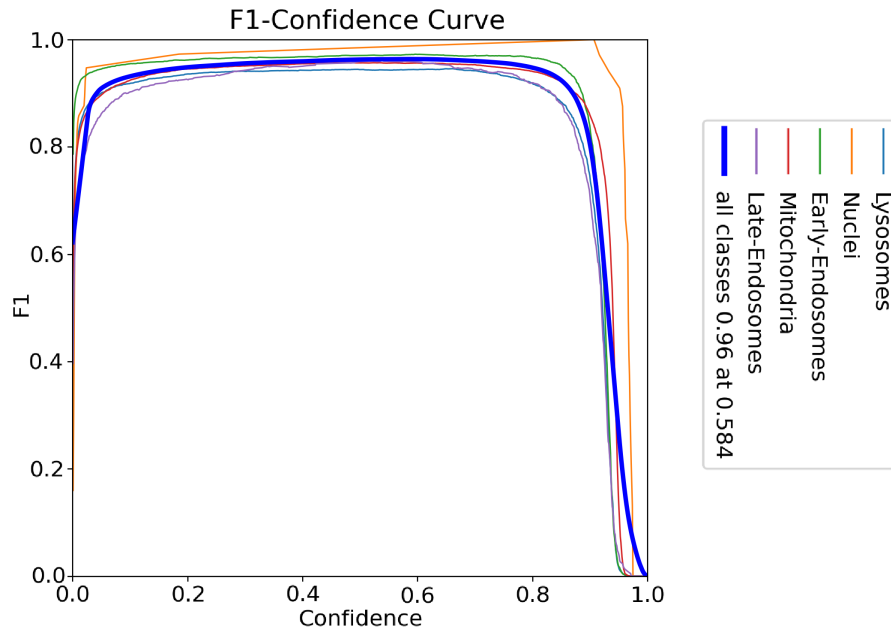


Figure 4.4: F1-Confidence Curve of Validation of Detection Task

Figure 4.5 relates the Precision with the Confidence of the model. As presented in Figure 4.3 Precision is the relation between True Positives and all predicted positives, achieving its maximum value when the number of False Positives approaches a null value, equating to a Precision of 1. The Precision-Confidence Curve obtains that value for several Confidence values and plots them, with the ideal curve possessing the Precision at a unitary value for all Confidence thresholds. From Figure 4.5, it can be perceived that the model surpasses 80% of Precision at low Confidence, which translates to it predicting more True Positives than False Positives. Precision, however, does not account for the number of existing True Positives or False Positives, which makes it an unreliable measure if used alone or not taking into consideration how many predictions were made and what they were.

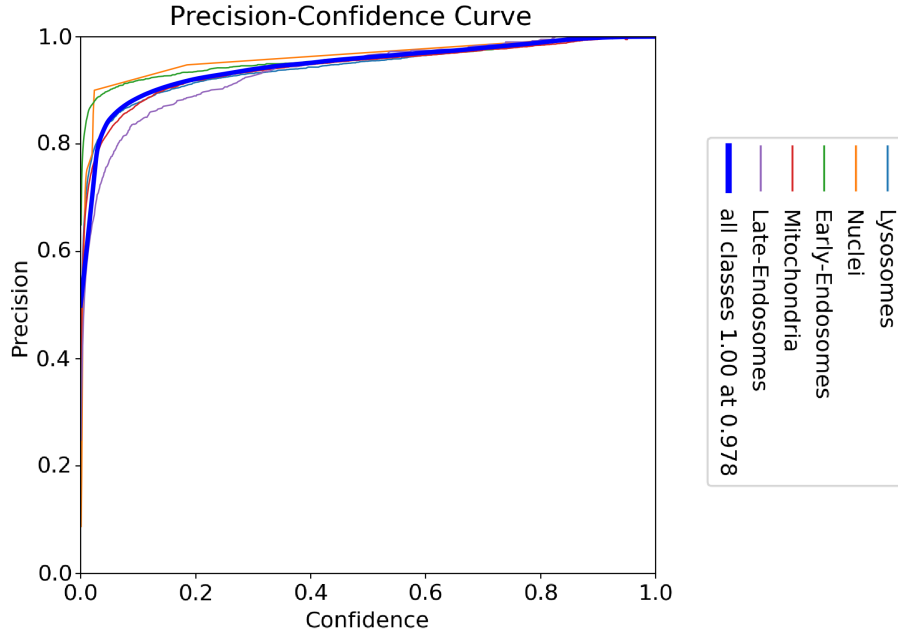


Figure 4.5: Precision-Confidence Curve of Validation of Detection Task

For the test set, the best parameters were used. These parameters were obtained by saving the training state every 10 epochs with the addition of a "best.pt" file that saves the parameters that provided better results. Test results are grouped in Table 4.2, being relatively high considering YOLOv8 is not a model mainly used for these datasets, as it was trained with the COCO dataset, known to be colorful and with high contrast [42]. The obtained results reflect a high Recall, high Precision and high F1 score, recommending the model's usage for the task.

Table 4.2: Test set results for confidence threshold of 0,5 of Detection Task

Recall	Precision	F1-Score
0,938	0,908	0,922

Visual results representing identification examples of organelles with Nuclei, Mitochondria, Early-Endosomes, Lysosomes, and Late-Endosomes, together with associated confidence, can be observed in Figure 4.6.

Overall, the results obtained from the training and testing of the YOLOv8 detection model demonstrate a strong and consistent performance. The quantitative evaluation shows that Precision, Recall and F1-Score all achieved values equal to or above 0.9, indicating a high level of accuracy in both detection and localization tasks. These metrics suggest that the model is effective at minimizing false detections while successfully identifying the majority of relevant objects. Furthermore, qualitative visual analysis corroborates these findings, as the model was able to correctly detect most

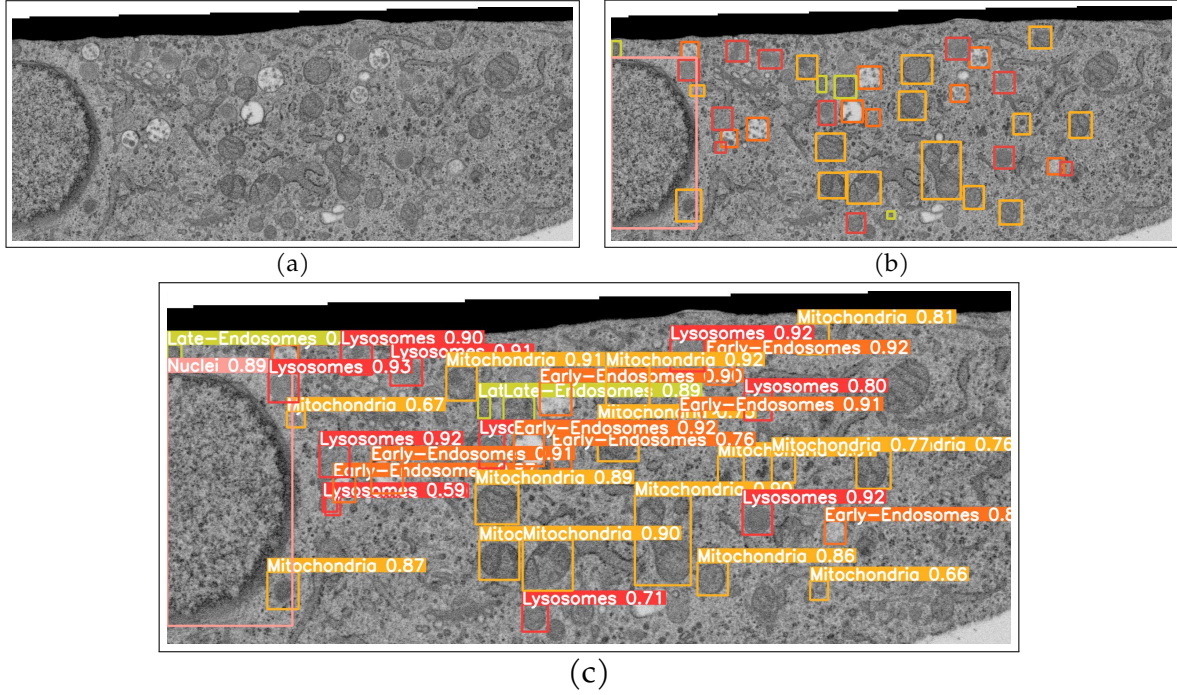


Figure 4.6: Detection of organelles done by YOLOv8 in Detection Task. (a) is the original image to be analysed; (b) is the location of the organelles with Nuclei in pink, Mitochondria in yellow, Early-Endosomes in orange, Lysosomes in red and Late-Endosomes in green; and (c) is the prediction made by YOLO to detect the organelles and respective confidence

target regions within the images, producing predictions that closely align with the ground truth. Together, these results confirm the robustness and reliability of the trained model, supporting its suitability for the intended application and solidifying YOLOv8's employment in segmentation tasks. The main limitation challenging this task was the short amount of data, forcing the exploitation of FIB-SEM technology to obtain more data from the volume and random split used to separate the data across the train, val and test sets, which may have created overestimated results by splitting similar sections for training, validating and testing.

4.2 YOLO Segmentation with Data Augmentation and Multi-Plane Sampling

Having acknowledged that YOLO can perform image detection on these types of images, preparing the dataset for image segmentation becomes the most logic step to follow. This section will provide a detailed path through the process by presenting the algorithms followed to achieve a compromise between the YOLO argument *imgsz* and the size of the used images, while also increasing the number of instances per class by employing Data Augmentation. Since 1778 slices proved to be a low number for the

planned experiments, an important aspect of FIB-SEM technology was used to obtain more images from the ones provided.

General Preparation of the Dataset

Starting the segmentation of the organelles one issue proved to be an obstacle: allocated memory. Out Of Memory errors, which occurred on a Intel i7-8700, 32 GB, NVidia GTX 1070 OC, are very common in YOLO training and stem from the model requiring more memory to be allocated than the device can provide. Due to the image segmentation model needing more points in the labels to form the boundaries of the organelle to be segmented, it needs more calculations, increasing the time and memory needed from the GPU. Factors that contribute to more memory being allocated include but are not limited to image size, batch size, model size (see Table 4.1 and Table 4.4) and, reiterating, the increase of points in the labels. Since both masks and original images have dimensions of 2023 pixels in width and 846 pixels in height they can be sliced into 3 images of 512 pixels by 512 pixels. The small dimensions will offer more room to avoid the Out Of Memory error and, if allowed, be filled with the demand of an increased batch size and larger model. These new image dimensions are obtained, turned into new labels, and segmented again with theoretically less memory needed for allocation. This change will however have an impact on one of the organelles. The nuclei, whose dimensions are greater than 512 pixels in height as seen in Figure 3.4, will be excluded from the segmentation.

Figure 4.7 represents the original image and its 3 resulting new dimensions images. Once the masks receive the same dimensioning they are to be subjected to pre-processing algorithms for extracting the labels in polygon format. The three main ones are a "mask check", which checks if there is at least one white pixel on the image, discarding the fully black images. This algorithm makes sure the model does not use useless images and guarantees that no image is entirely composed of background information; the "black contour" removes areas of white pixels that have pixels on the image border. This algorithm ensures the organelles are intact in the training section as to not confuse the model and lower the quality of the results; the "area check" checks for blobs that are too small to be a known organelle and deletes the image if the biggest area in the image falls in that criteria. These algorithms are to be interleaved since "black contour" could originate a black image.

Multi-Plane Sample Extraction

FIB-SEM images are a spliced section of a parallelepiped sample volume, as demonstrated in Section 3. This means that the same volume can be analysed in the direction perpendicular to 3 unique faces, meaning that in addition to the original axial XY slices, orthogonal slices were generated along the XZ and YZ planes. This approach exploits

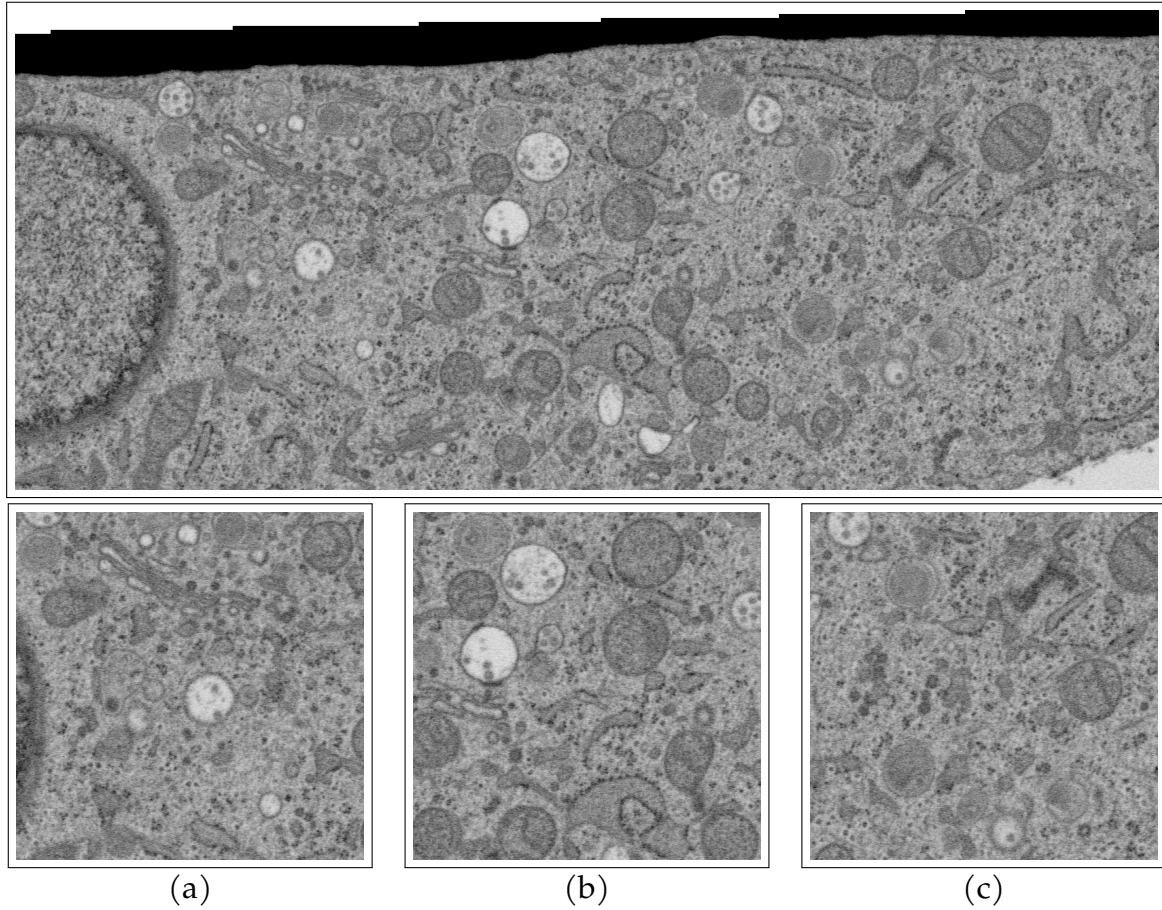


Figure 4.7: Example of an image sliced into 3 512x512 images. (a) is the center-left 512² pixels image, (b) is the group of the pixels in the center of the original and (c) is comprised of the immediate right pixels next to the center

the isotropic or near-isotropic resolution of the FIB-SEM volume and allows different cross-sectional views of the same underlying structures to be included in the dataset. As a result, multiple complementary representations of the same anatomical or material features were obtained without introducing synthetic content. So, from the 1778 images of dimensions 2023 by 846 pixels, an additional 2023 images of dimensions 1778 by 846 pixels and 846 images of dimensions 2023 by 1778 pixels were created. An example of the result can be found in Figure 3.1. With the images acquired, the same protocol applied to the original images can be used. The images of dimensions 1778 by 846 pixels can also produce 3 sections of 512 by 512 pixels and the images of dimensions 2023 by 1778 pixels can produce 9 sections. This means that the original 2023 by 846 images result in $1778 \times 3 = 5334$ images, the new 1778 by 846 images result in $2023 \times 3 = 6069$ images and the new 2023 by 1778 images result in $846 \times 9 = 7614$ new images, totalling $5334 + 6069 + 7614 = 19017$ images of dimensions 512 by 512. However, this is not the achieved number because of how the FIB-SEM volume was obtained.

Figure 4.8 shows the extremities of the newly created images. Figures 4.8.(a) and 4.8.(b) refer to the first and last images of the set extracted from the X plane, while

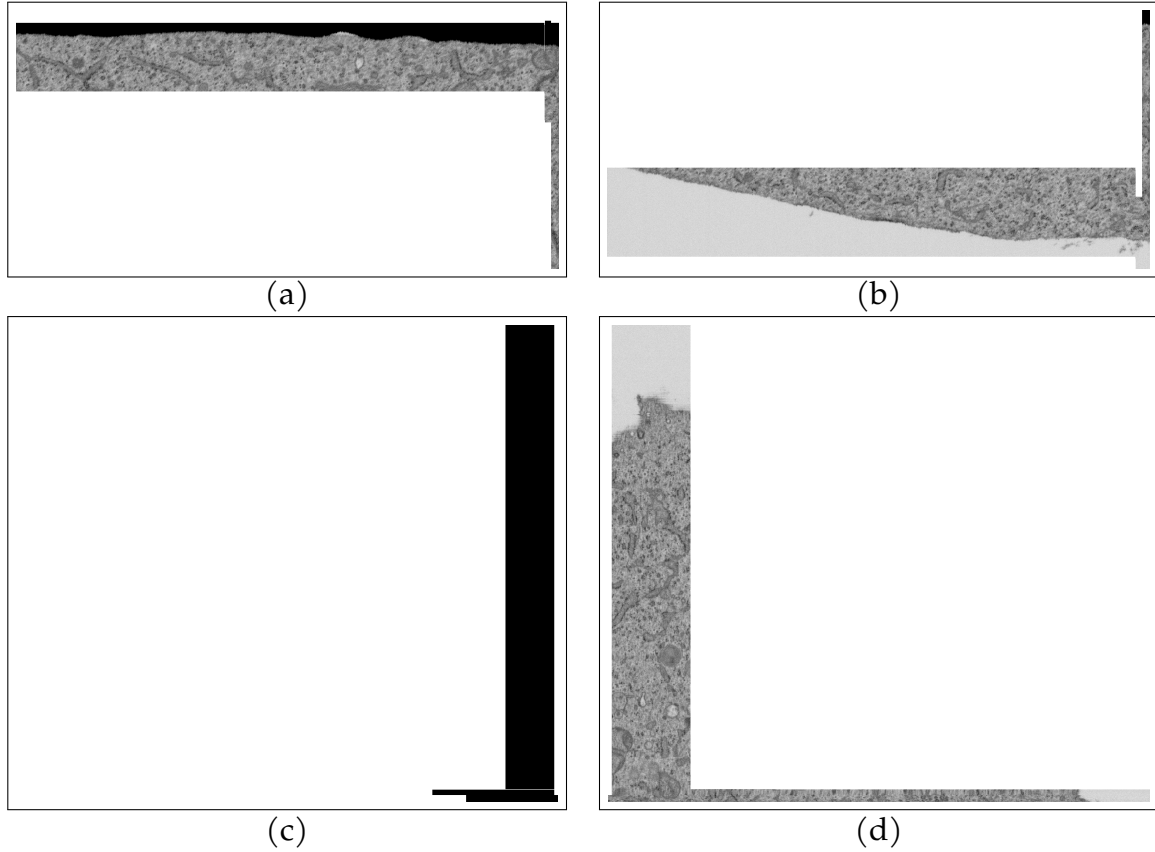


Figure 4.8: Extremities of the volume when analysed from the X plane ((a) and (b)) and from the Y plane ((c) and (d))

Figures 4.8.(c) and 4.8.(d) refer to the first and last images of the set extracted from the Y plane. The main shared characteristic among these images is a large white area covering most of the images. One line from Figure 4.8.(c) can be seen in Figure 4.7 on the large image at the top, while the others are not perceptible until further down the original dataset. The white area extends into a few other images of the new sets, removing the first 14 and the last 15 images from the X plane and the first 130 and the last 245 images from the Y plane. This results in a new total of 1995 images obtained from the X plane and 471 images from the Y plane, which when converted to sections of 512 by 512 pixels result in $1995 \times 3 = 5985$ images from the X plane and $471 \times 9 = 4239$ images from the Y plane.

Class-Wise Data Augmentation

One problem encountered from learning experiments stemmed from the fact that the classes were unbalanced. Figure 3.6 exemplifies the visual differences from each class which sets a precedent to how they behave when subjected to the segmentation model. However, the difference extends to the instance count of the classes. A numeric count of each instance from the 512 by 512 pixel images was conducted to assess how each class was to be augmented.

Table 4.3: Absolute and relative count of each instance of each class

	Plane X	Plane Y	Plane Z	Total
Mitochondria	6699 (42%)	8162 (36%)	8091 (43%)	22952 (40%)
Lysosomes	3373 (21%)	5510 (24%)	3835 (20%)	12718 (22%)
Early-Endosomes	3963 (25%)	6742 (30%)	4793 (26%)	15498 (27%)
Late-Endosomes	1873 (12%)	2164 (10%)	2015 (11%)	6052 (11%)
Total	15908	22578	18734	57220

Table 4.3 categorises how many instances there are per class per plane. The count for Mitochondrion objects is higher above all, pointing to little augmentation done to this class. Lysosomes and Early-Endosomes possess a relatively close number of instances, subjecting themselves to a similar degree of augmentation. Late-Endosomes, however, possess a very low count, with the overall total not being enough to compare to a single plane of Mitochondrion objects, encouraging a more severe degree of augmentation.

The library SciPy [49] was used to perform the augmentation operations. SciPy is an image manipulation library that can perform pixel-wise operations to an image. Pertinent to this section, it also transforms an image by means of rotations, reflections, enhancements, Gaussian blurs and others. To maintain a faithful relationship to the organelles only rotations and reflections were employed. Figure 4.9 showcases the augmentations done on both original images and the respective masks.

With 3 operations and the possibility to add more through 90 degree rotations, 45 degree rotations and other operations maintaining fidelity to organelle volume, the sample size can be increased to acquire more data to the model. Applying each set of operations according to plane, Mitochondria needed no augmentations; Lysosomes needed 3 extra operations to the X plane, 0 to the Y plane and 0 to the Z plane; Early-Endosomes needed 1 extra operation to the X plane, 0 to the Y plane and 1 to the Z plane; and Late-Endosomes needed 3 extra operations to the X plane, 3 to the Y plane and 2 to the Z plane. Post augmentation, the total count for the organelles was 22952 Mitochondria instances, 22837 Lysosomes, 24254 Early-Endosomes and 22193 Late-Endosomes.

Argument Settings and Training

The model used for the segmentation was YOLOv8x-seg. According to Table 4.4, also obtained from Shaoni Mukherjee’s tutorial [46], this model possesses better Mean Average Precision, or mAP, a metric to be explained later on, and the architecture of the model uses more parameters, requiring more floating point operations per second which requires more computational power and in turn more time to train, but what is not a limitation for this task.

The model was trained with an image size of 512 to match the dimensions of the dataset, once again ensuring the dimensions of the organelles stayed the same throughout the

Cell Segmentation in FIB-SEM Microscopy Images

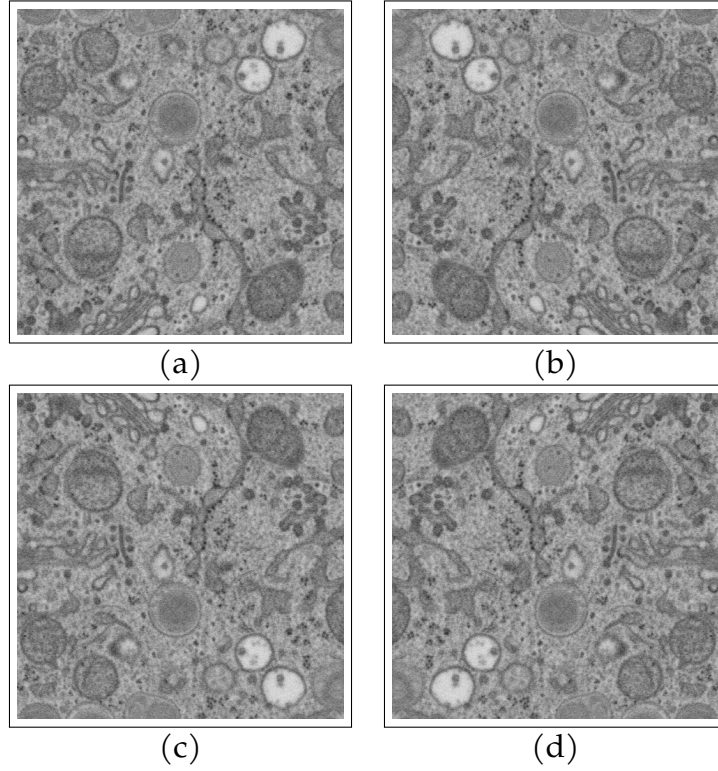


Figure 4.9: Augmentations performed on the images and masks. (a) is the original 512 by 512 pixel image, (b) is a reflection across the y axis, (c) is a reflection across the x axis and (d) is a 180 degree rotation.

Table 4.4: Difference among the segmentation models of YOLOv8 [46] on COCO set

Model	size (pixels)	mAP ^{box} 50-95	mAP ^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

training. The batch size used was 8 images per iteration, which was possible to reach in spite of the high demand model due to the smaller image size. Like the model trained for the detection task, this model was trained for 100 epochs, the default value. Other arguments were set to their default value.

Results

With the segmentation task there are now two sets of graphs, one relating to the bounding boxes which were also a product of the detection tasks, and relating to the segmentation masks, which are the pixel areas obtained from the segmentation task. The mask-related metrics are presented in this section.

Figure 4.10 presents the F1-Confidence curve. According to the curves the F1-Score dipped after the 50% Confidence threshold, hitting a peak of 0.77 at only 0.258 confidence. This could mean that the model was able to predict on the classes at a lower confidence. To prove this more curves must be analysed.

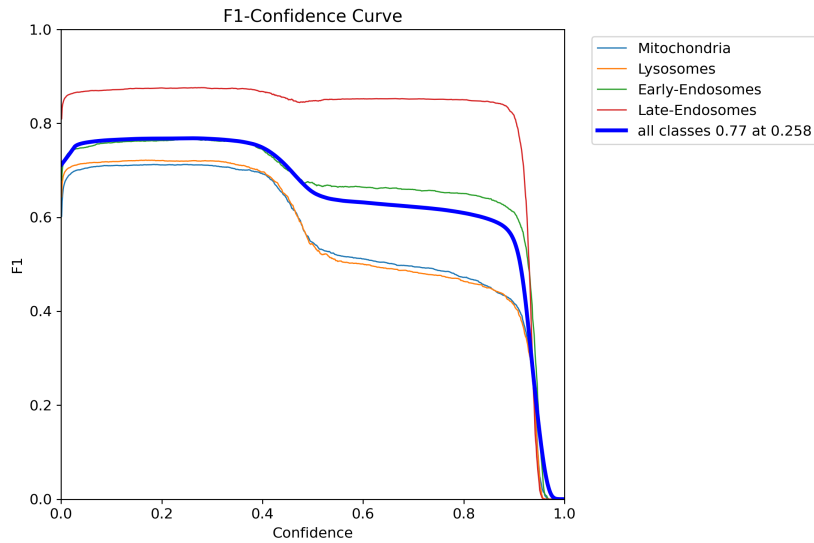


Figure 4.10: F1-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task

Figure 4.11 presents the Precision-Confidence curve of the trained model. It shows that Precision peaks at 0.97 Confidence with an unitary value. It presents two plateaus of values on similar spots to the F1-Confidence curve with different values. A lower average Precision under 50% confidence indicates that the model was performing more predictions on the images, while containing many true predictions, only around 65% of them were correct. A higher value of Precision after the middle point means that either not so many predictions were being made, or the model was able to recognize where most of the true positives were. To get more data on it the Recall-Confidence

curve should be analysed.

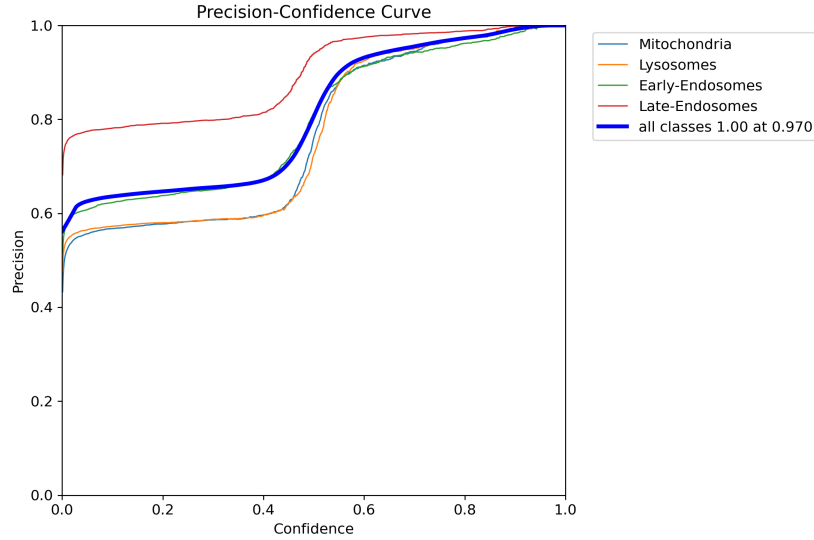


Figure 4.11: Precision-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task

Figure 4.12 presents the Recall-Confidence curve of the trained model. Contrary to the Precision-Confidence, it maxes at 0% confidence with almost Recall's max value. It's high beginning indicates that at lower confidence most, if not all, of the true values were predicted to be the true values. This correlates to the previous curves explanation that the model was performing a lot of predictions, whether they were correct or not. On the other side of the 50% confidence threshold, a lower Recall overall value indicates that the model could not perform as many predictions, meaning that the high Precision comes from lower amount of predictions.

Figure 4.13 presents the Precision-Recall curve, a plot used to more accurately explore the relationship between predictions, portrayed by Precision, and actual values, portrayed by Recall. The Figure shows how high Precision meets low Recall and vice-versa, with a dip between 40% Recall and 65% Precision. This line lowers the Mean Average Precision of the model at an Intersection over Union of 50% to 0.834. Mean Average Precision [50], or mAP, is a metric used to evaluate a Deep-Learning model by performing an arithmetic average to the Precision values at several levels of Confidence, illustrating how Confidence affects the Precision of the model. Also known as "The Area Under The Curve", it can be calculated by determining the integral of the Precision-Recall curve.

To better understand how many predictions were being made and how each class is predicted on, as was used in the detection experiment, Figure 4.14 contains the Confusion Matrix computed from the validation of the model. It can be understood that over 90% of ground truth instances were correctly predicted to be true values from

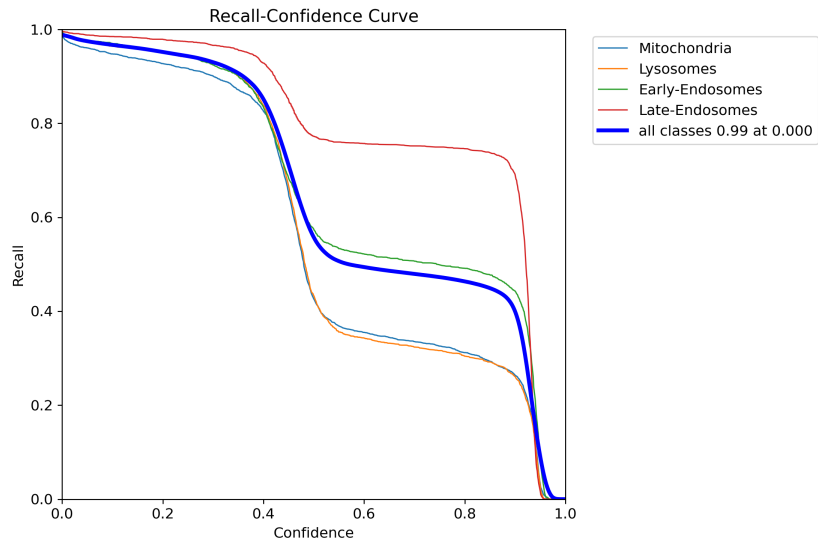


Figure 4.12: Recall-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task

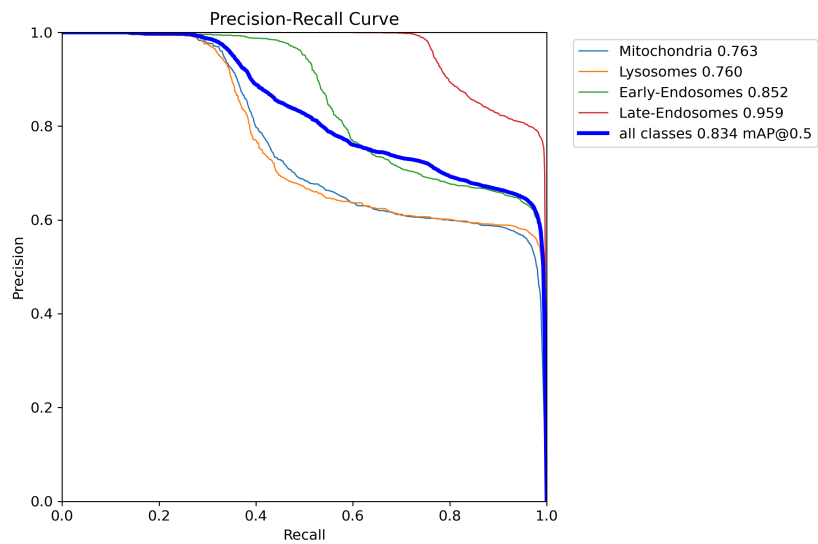


Figure 4.13: Precision-Recall Curve for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task

Cell Segmentation in FIB-SEM Microscopy Images

each class. This allowed the model to determine that under 10% of instances of each class were not predicted on, being classified as background in the matrix. On the right side it shows that up to 40% of background was classified to be a designated organelle, hinting that the model is not entirely capable of fully discerning some organelles from others.

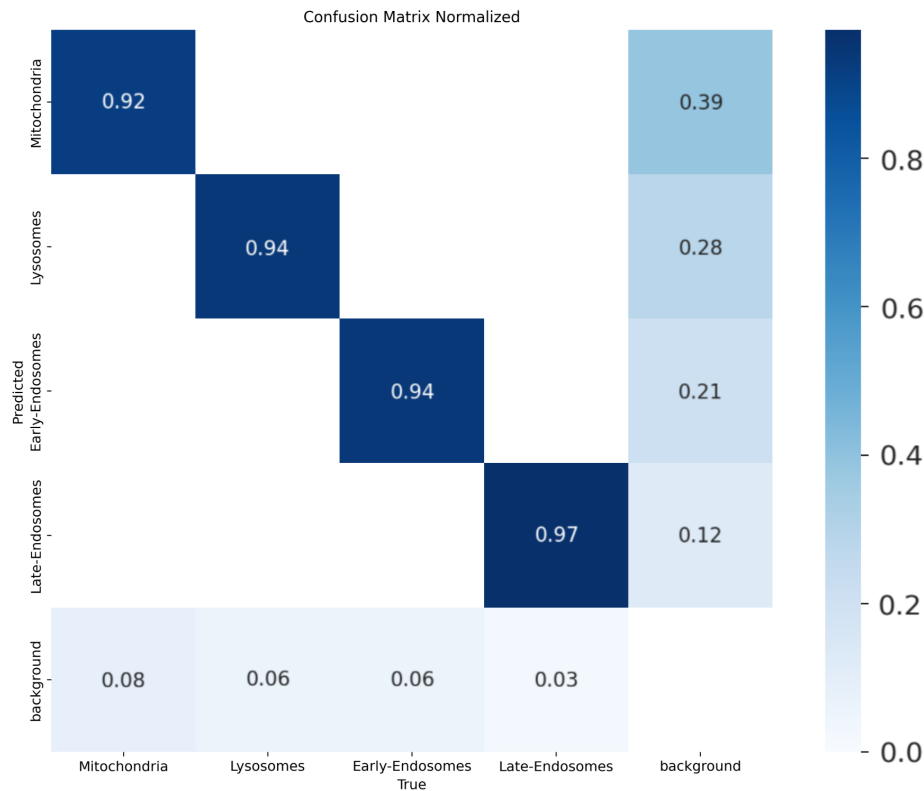


Figure 4.14: Normalised Confusion Matrix for Validation for Data Augmentation, Multi-Plane Sampling Segmentation Task

For the test set, the best parameters were used, saved on the "best.pt" file. Test results are grouped in Table 4.5, which although lower than the values obtained from the detection task are still high for FIB-SEM images. The obtained results reflect a high Recall, higher than average Precision and high F1 score, translating to the model being able to get almost all true values correctly predicted, most predictions being correct. These values indicate that data augmentation and attention to fidelity towards the organelles proved to be satisfactory decision.

Table 4.5: Test set results with confidence threshold of 0,5 for Data Augmentation, Multi-Plane Sampling Segmentation Task

Recall	Precision	F1-Score
0,945	0,651	0,771

Visual results representing identification examples of organelles with Mitochondria, Early-Endosomes, Lysosomes, and Late-Endosomes can be observed in Figure 4.15.

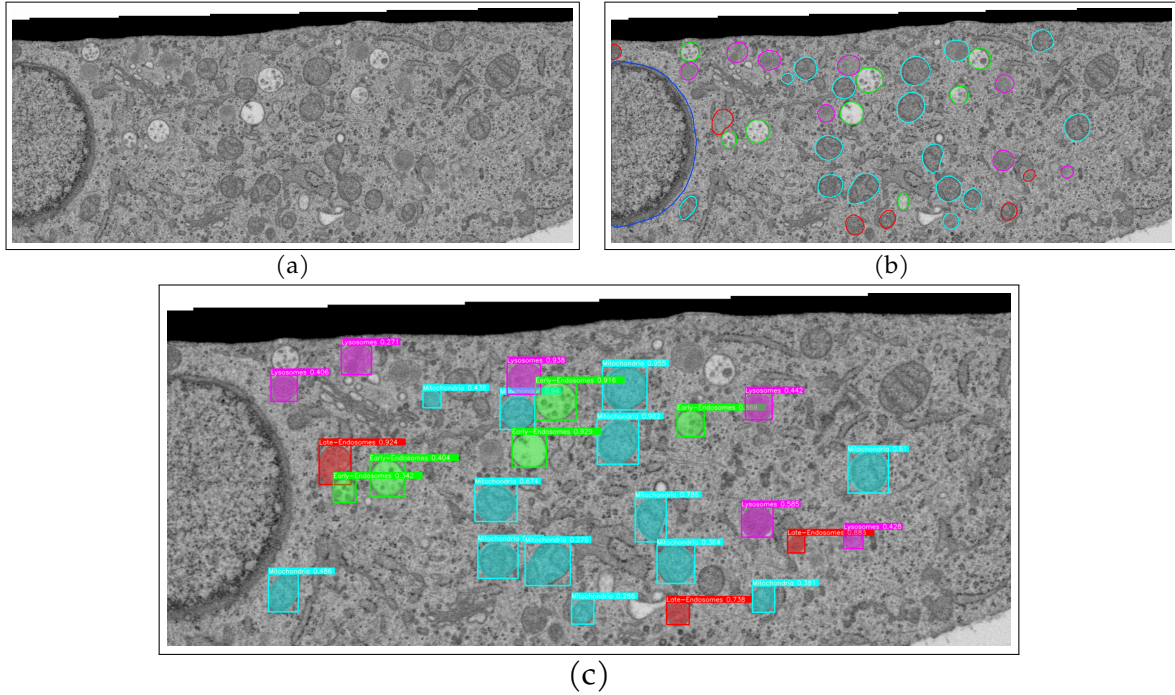


Figure 4.15: Detection of organelles done by YOLOv8 for Data Augmentation, Multi-Plane Sampling Segmentation Task. (a) is the original image to be analyzed, (b) is the location of the organelles with Mitochondria in blue, Lysosomes in pink, Early-Endosomes in green, and Late-Endosomes in red, and (c) is the prediction made by YOLOv8 to detect/segment said organelles and respective confidence.

Overall, the results obtained from the validation and testing of the YOLOv8 segmentation model demonstrate the model's reliability for the task. Although a confidence level of 50% separated two distinct results the quantitative evaluation indicated values higher than 60% for all test metrics, indicating a modest level of accuracy in both detection, already concluded in the detection task in the previous section of this chapter, and segmentation. These metrics suggest that the model makes more predictions with lower confidence and more correct predictions with higher confidence, with an accentuated slope on the curves caused by a confidence level of 0,5 attributed to predictions indicating that the predictions are reduced more acutely after 0,5 confidence, lowering Recall by failing more of the ground truth but increasing Precision by being more exact. Qualitative evaluation corroborates this statement with fewer predictions compared with the detection task, with confidence values balanced throughout the segmentation prediction. Together, these metrics point towards consistency of YOLOv8 as a segmentation model for low contrast, FIB-SEM images.

4.3 YOLO Segmentation with Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

The segmentation model obtained from the previous section exhibited very good behaviour when presented with these types of images. However, compared to other models derived from similar experiments in other/similar fields, this model begins to reveal a potential underlying issue on the high metrics obtained. The work done by Zini Jian et al. [51], for instance, hints at it. They developed a Deep Learning model with YOLOv8 as its backbone to perform a detection task on low contrast noisy images, with the model achieving a mean Average Precision value at a 50% IoU threshold of 70,21%. This means that a model made from YOLOv8 specifically for a detection task underperformed compared to one trained with only YOLOv8 with most of its parameters set to their default values which, although possible, seems not accurate. After a thorough investigation it was revealed that the dataset may have been created in a not completely fair way, resulting in data leakage. For the past experiments the dataset was being randomly separated into the train, validation and test set, meaning that some images in the train set could very similar to the ones in the validation set and the test set. This is because, taking into consideration the fact that because they were extracted from the same volume image n was very similar to images $n - 1$ and $n + 1$ in different sets. That means that, to obtain a more accurate model the dataset separation phase must be more strict in avoiding randomization. In addition, the Data Augmentation performed resulted in similar images to the originals being created only noting the augmented class. To combat this Data Augmentation will be done by YOLOv8 training, since it is flexible in which augmentations are performed and thus some can be disabled or restricted.

During a review of the YOLOv8 documentation, particular attention was given to hyperparameter optimization, which aims to identify the most effective configuration of arguments for training the model. Among the parameters examined, several were found to modify the input images in various ways. These image-altering parameters warrant careful consideration, as their impact may be significant depending on the characteristics of the dataset under study.

Preparation of the Dataset

As stated, in the previous sections the dataset was being randomly split across the YOLOv8 dataset tree. This, coupled with an increase in data through augmentation, generated a less efficient model. So, the dataset would first be analysed from the 2 other planes X and Y, resulting in the aforementioned 2023 images of dimensions 1778 by 846 pixels in the X plane, 846 images of dimensions 2023 by 1778 pixels in the Y plane and 1778 images of dimensions 2023 by 846 pixels in the Z plane, utilizing the same protocol, with X and Y being the new planes. The same strategy of acquiring new square

images with sides of 512 pixels was also implemented, aiming to train the model as faithfully to the original size of the organelles as possible. With the extra care in place to avoid similar images in different sets and ensuring the model was being validated and tested on the original dataset to avoid increased noise levels, planes X and Y were sent to be the training set. Additionally, the first and last group of images similar to the ones in Figure 4.8 were removed, bringing the X dataset to $1995 \times 3 = 5985$ square images and the Y dataset to $471 \times 9 = 4239$ square images, used as the training set. The validation and test set utilized the Z Plane, with a section of images in the validation-test boundary enacted to lower the chance of similarity among the sets. Table 4.6 adds a visual aid to the split of the dataset.

Table 4.6: Dataset separation into YOLOv8 sets

	Train	Val (Z Plane)	Boundary (Z Plane)	Test (Z Plane)
Images in Sets	Planes X and Y (9928)	2454	426	2454

This segregation certifies that the model will be validated and tested solely on the original data, leaving the training set with images containing structures not present in those sets. This approach ensures that Data Augmentation is applied exclusively to the training set, thereby increasing data variability while preserving the integrity of the validation and test datasets.

Argument Settings through Hyper-parameter Tuning

The question raised in the previous section ignited more inquiries on how other arguments, so far set to their default value or changed without detailed analysis, would influence the training and validation of the model. The search for the best values to be placed in the training function is called Hyper-parameter Tuning where the goal is to arrange various combinations of arguments, train the model with them, catalogue the results and repeating the process until the results are satisfactory.

Hyper-parameter optimization was not limited to training arguments alone. As previously mentioned, the image dimensions of the dataset were also evaluated to determine whether a reduced, square resolution yielded better performance than using the full image. Furthermore, as shown in Table 4.4, five YOLOv8 segmentation models, ranging from "yolov8n-seg.pt" to "yolov8x-seg.pt", were considered to assess whether the largest and most complex model provided superior performance for the task at hand.

Table 4.7 summarizes the considered Hyper-Parameters, their relationship with YOLO and their range values. *Model* is the model used for the task, exploring every model introduced in Table 4.4. *Batch* is the batch size, the argument responsible for the amount of images to train at once, which will attempt to use 32, 16 or 8 images per iteration,

falling back on a negative one, the value to be set by YOLOv8 to determine the best batch size to avoid running out of memory. The variable lr_0 represents the initial learning rate, which governs how the model updates its internal parameters during the early training epochs, and is configured to fall within the range specified by the YOLOv8 documentation [52]. The variable lrf is the final learning rate and will act on the initial learning rate by setting it to a percentage of its value from 10% to 100%. Lastly, $warmup_epochs$ stands for the number of epochs where the initial learning rate will act before being decreased by the final learning rate, with a minimum of 0 epochs, meaning the learning rate will be constant and equal to $lr_0 * lrf$, and a maximum of 5 epochs.

Table 4.7: Hyper-parameter Tuning Settings

Variable	Type (YOLOv8)	Range
model	Model	n; s; m; l; x
batch	Argument	-1; 2; 4; 8; 16; 32
lr_0	Argument	(1; 2; 3; 4; 5; 6; 7; 8; 9; 10) $\times$ 0,001
lrf	Argument	(1; 2; 3; 4; 5; 6; 7; 8; 9; 10) $\times$ 0,1
$warmup_epochs$	Argument	0,0; 1,0; 2,0; 3,0; 4,0; 5,0

Since computational resources were limited, preventing all the variables from being tested at once due to an increased time period, one variable was tested on per turn, with the exception to the model and batch which were combined on the same experiments because the impact on time was not so severe. With this, the three best models trained under model and batch are presented in Table 4.8, with names like Modelx for convenience to the report. These models were chosen because Model1 showcased the best mAP at 50% IoU threshold and Recall, meaning it is competent at detecting and segmenting, missing a few objects, Model2 had the best mAP for higher IoU threshold for bounding box detection and precision, demonstrating its detection prowess, and Model3 had the best mAP for higher IoU threshold and precision for mask segmentation, being more suited for segmentation.

Table 4.8: Best models for the first combination of variables

Model Name	model	batch	mAP50	mAP50-95
Model1	yolov8m-seg.pt	-1	0,67	0,42
Model2	yolov8l-seg.pt	-1	0,66	0,42
Model3	yolov8s-seg.pt	-1	0,64	0,46

Introducing these three models to a ranging initial learning rate produced the model in Table 4.9. Model4 was the only model to perform better than other models in every metric apart from mAP at higher IoU threshold for mask segmentation, which improved in relation to Model1.

Table 4.9: Best model for the second combination of variables

Model Name	model	batch	lr0	mAP50	mAP50-95
Model4	yolov8m-seg.pt	-1	0,06	0,70	0,43

The same protocol was applied with the addition of the final learning rate, resulting in Table 4.10. Model5 was able to maintain a high mAP at 50% and higher IoU threshold and improve Recall for both bounding box detection and mask segmentation, with a slight dip in Precision, meaning that true values were being predicted more but less predictions were made correctly.

Table 4.10: Best model for the third combination of variables

Model Name	model	batch	lr0	lrf	mAP50	mAP50-95
Model5	yolov8m-seg.pt	-1	0,06	0,3	0,69	0,44

Table 4.11 concludes the Hyper-parameter tuning with warmup_epochs being set to its maximum value. Model6 kept a high mAP for all IoU thresholds and improved Precision but achieved a lower Recall value. This means that predictions were now more accurate but did not predict on as many true masks.

Table 4.11: Best model for the fourth combination of variables

Model Name	model	batch	lr0	lrf	warmup_epochs	mAP50	mAP50-95
Model6	yolov8m-seg.pt	-1	0,06	0,3	5,0	0,69	0,44

The metrics obtained from Model6 are displayed in Table 4.12. The letters B and M are respective to Bounding Box and Mask and the numbers coupled with the mAP are the IoU thresholds. It demonstrates that mask wise mAP values at higher IoU thresholds are similar to the ones presented by YOLOv8 in Table 4.4. Additionally, several other metrics achieved above-average values, indicating that these parameter combinations, and the order in which they were applied, could still lead to a well-performing model.

Table 4.12: Metrics of the best model configuration obtained through Hyper-parameter tuning

mAP50 (B)	mAP50 (M)	mAP50-95 (B)	mAP50-95 (M)	Precision (B)	Precision (M)	Recall (B)	Recall (M)
0,66	0,66	0,47	0,42	0,59	0,58	0,68	0,67

Taken together, these findings support maintaining the dataset at its original image dimensions. The rescaling introduced by YOLOv8's `imgsz` parameter did not significantly alter the visual characteristics of the images, indicating that image resizing alone does not substantially influence the model's learning capability or performance outcomes in this context.

Argument Settings and Training

As extrapolated from image dimensions, the argument `imgsz` was set to 512, resizing all input images to a square with dimensions of 512 by 512 pixels, thus maintaining organelle size.

YOLOv8 documentation noted that the model practiced its own form of Data Augmentation by applying rotations, zooms, colour changes and several others [53]. To ensure the model trained only on the provided data, the documentation also provided exactly which arguments altered the images and an explanation on their behaviour. Table 4.13 presents the arguments and their values to prevent YOLO from using excessive augmentations on data. Among the available arguments, `mosaic`, `hvs` arguments, `scale`, and `translate` are the most likely to negatively affect the training images because of the chance of leaving important and complementary information out of frame and changing the channels of the RGB images, losing the status of gray scale and with it more information. Arguments like `erasing` and `auto_augment` are only applicable to YOLO classification tasks, another branch of the YOLO family, but were still deactivated to ensure success in training.

Table 4.13: YOLO augmentation arguments values and functions

Argument	Value	Functionality
<code>hvs_h</code>	0.0	Alters the Hue (Colour) of the images
<code>hvs_v</code>	0.0	Alters the Value (Brightness) of the images
<code>hvs_s</code>	0.0	Alters the Saturation (Colour Amount) of the images
<code>translate</code>	Default	Shifts image, often out of frame
<code>scale</code>	0.0	Resizes images, enlargings may push border data out of frame
<code>flip_lr</code>	Default	Flips the image from left to right
<code>mosaic</code>	0.0	Combines four images into one, maintaining image size
<code>erasing</code>	Default	Randomly erases portions of the image during training.
<code>auto_augment</code>	Default	Applies automated augmentation policies for classification.

With the data size, model and other arguments set according to results from past experiments, training could begin with 100 epochs, the default value.

Results

Analysing the Mask oriented results, Figure 4.16 displays the F1-Confidence Curve obtained from the validation set. It shows that the peak occurs at 0,199 confidence with an F1-Score of 0,67, suggesting the model assigns lower confidence values to correct predictions. To obtain more data other curves must be explored.

Figure 4.17 is the Precision-Confidence Curve, showcasing an increasingly higher Precision with a peak of 0,97 at a Confidence level of 1. This means that highly confident predictions are almost never wrong. The sudden low values at high confidence arise from lower to no predictions being made with high confidence, resulting in a Prediction

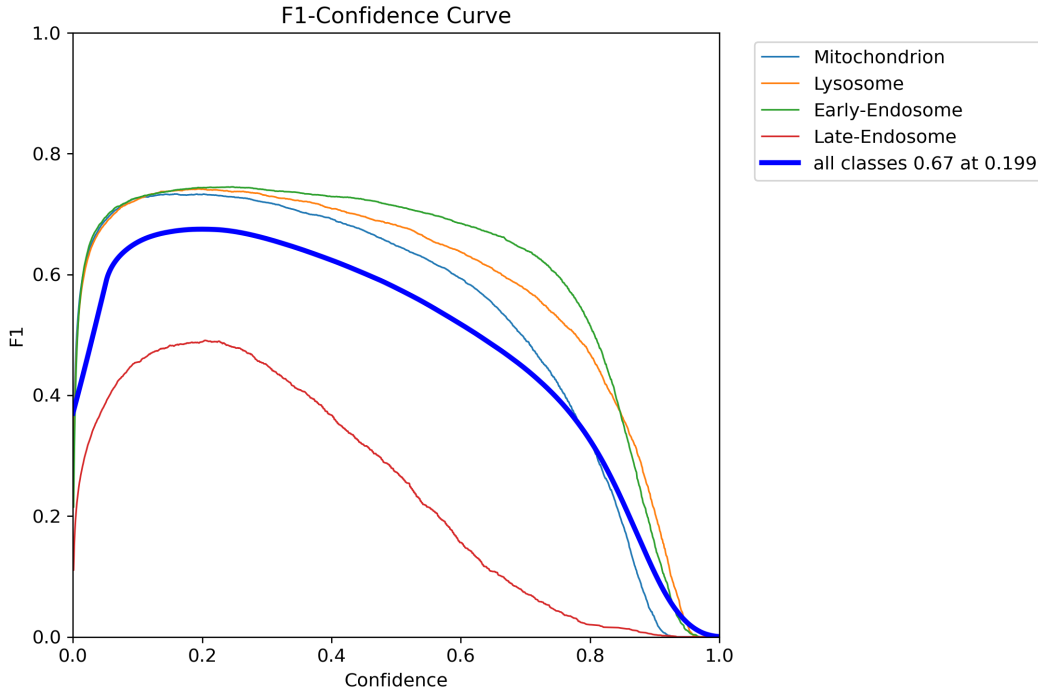


Figure 4.16: F1-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

value decreasing due to the predictions being incorrect, dropping to 0, and the uprise being calculated to be 0 True Positives divided by 0 predictions, which the algorithm is considering as 1. The overall high values, however, do not reflect on the F1-Score, meaning the dip at higher confidence levels stems from the Recall-Confidence Curve.

Figure 4.18 is the Recall-Confidence Curve. The first apparent contrasting feature to the Precision-Confidence Curve is its near inverse proportionality to the Confidence. It peaks at 0 Confidence level with a Recall value of 0,91. This result means that, to ensure most true instances are segmented, most predictions must be accepted. This is a hallmark of difficult segmentation targets, typical of FIB-SEM images. The F1-score Curve plays a role in the decision of the Confidence level for testing and prediction by relating both Precision and Recall, offering a common peak to decide the Confidence level.

Figure 4.19 is the Precision-Recall Curve, relating the two metrics. Mapping both metrics at the same Confidence values, it becomes apparent that when the Intersection over Union threshold is set to 50% the mean Average Precision reaches a value of 0,691. This curve explains that the model's ability to detect and segment is reasonably good.

Figure 4.20 is the Normalised Confusion Matrix, displaying exactly how the model predicts each class. It shows that much of Mitochondria are not predicted, revealing that the model is not fully capable of detecting them. Late-Endosomes are often classified as other classes or background, highlighting the similarity between the classes and

Cell Segmentation in FIB-SEM Microscopy Images

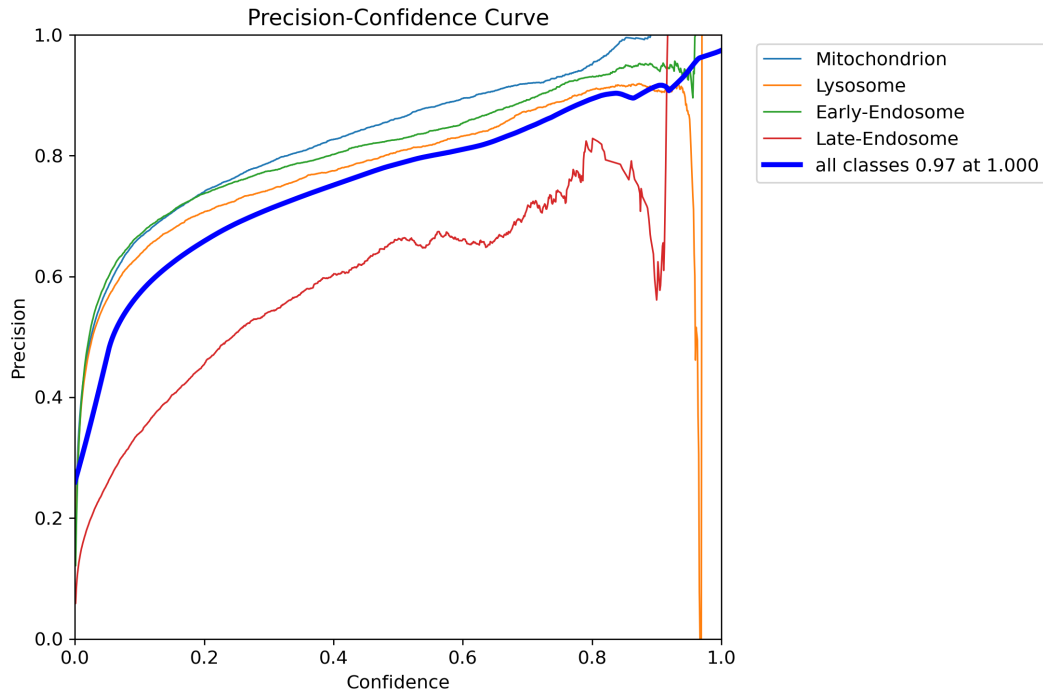


Figure 4.17: Precision-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

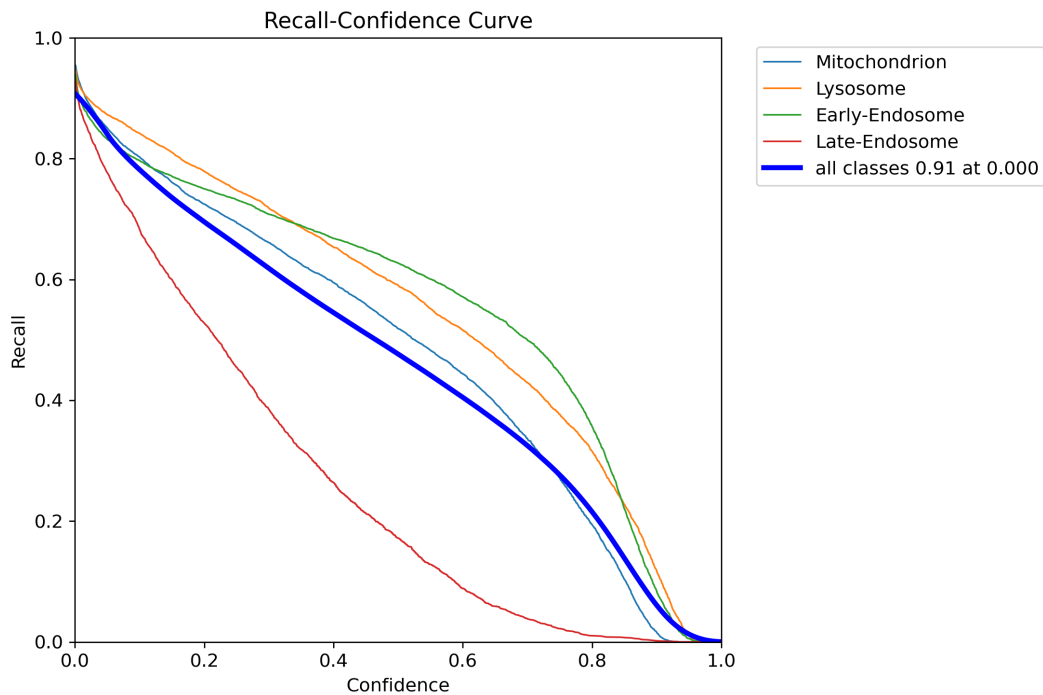


Figure 4.18: Recall-Confidence Curve for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

the need for class-level augmentation, but also indicating that Lysosomes and Early-Endosomes are the most unique of the classes.

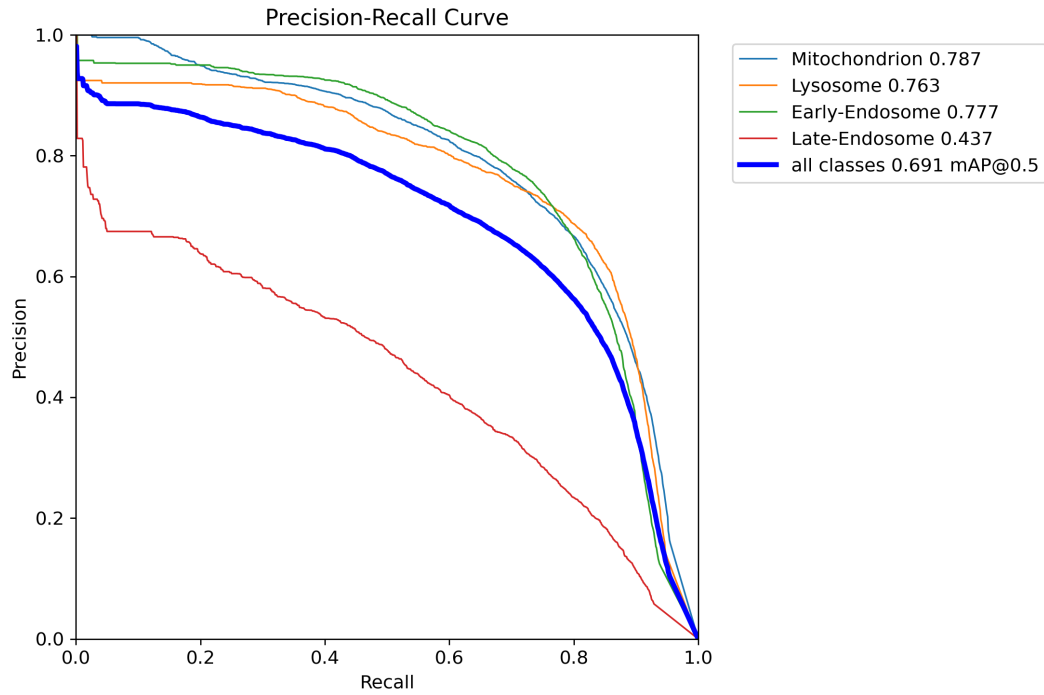


Figure 4.19: Precision-Recall Curve for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

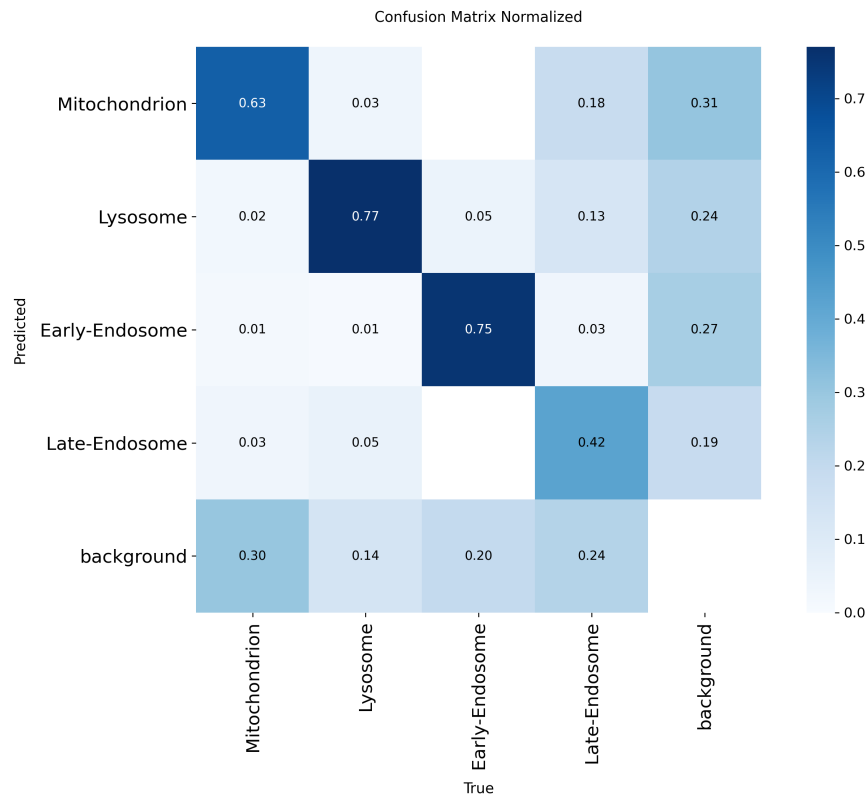


Figure 4.20: Normalised Confusion Matrix for Validation for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

For the test set, the best parameters were used, saved on the "best.pt" file. Test results are grouped in Table 4.14, which, although lower than the values obtained from the detection task, are still high for FIB-SEM images. The obtained results reflect a high Recall, higher than average Precision and high F1 Score, translating to the model being able to get almost all true instances correctly predicted and most predictions being correct. These values indicate that re-designing data augmentation, hyper-parameter tuning and attention to fidelity towards the organelles proved to be a correct decision.

Table 4.14: Test set results with confidence threshold of 0,2 for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

Recall	Precision	F1-Score
0,74	0,682	0,71

Visual results representing identification examples of organelles with Mitochondria, Early-Endosomes, Lysosomes, and Late-Endosomes can be observed in Figure 4.21.

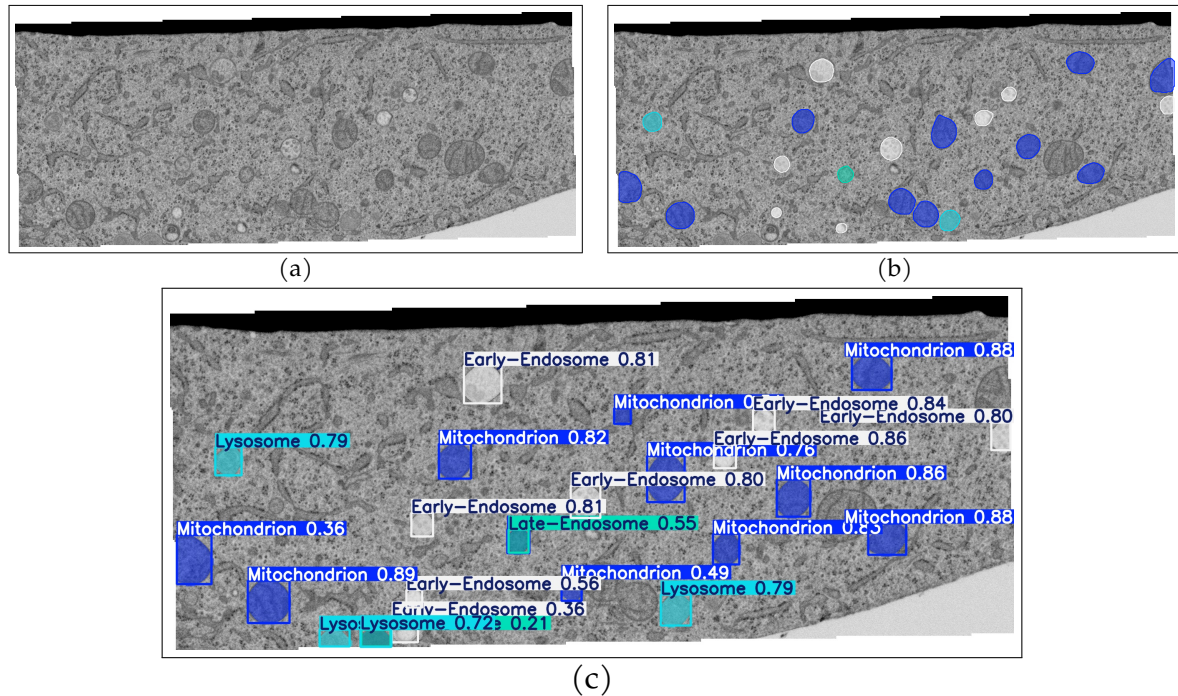


Figure 4.21: Detection of organelles done by YOLOv8 for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning. (a) is the original image to be analyzed, (b) is the location of the organelles with Mitochondria in blue, Lysosomes in cyan, Early-Endosomes in white, and Late-Endosomes in green, and (c) is the prediction made by YOLOv8 to detect/segment said organelles and respective confidence.

Analysing the mask-oriented test results, Figure 4.22 displays the F1-Confidence Curve obtained from the test set. It shows that the peak occurs at 0,183 confidence with an F1-Score of 0,71, being on par with the validation set.

Figure 4.23 is the Precision-Confidence Curve, showing an increasingly higher Precision with a peak of 1 at a Confidence level of 0,996. This means that predictions are

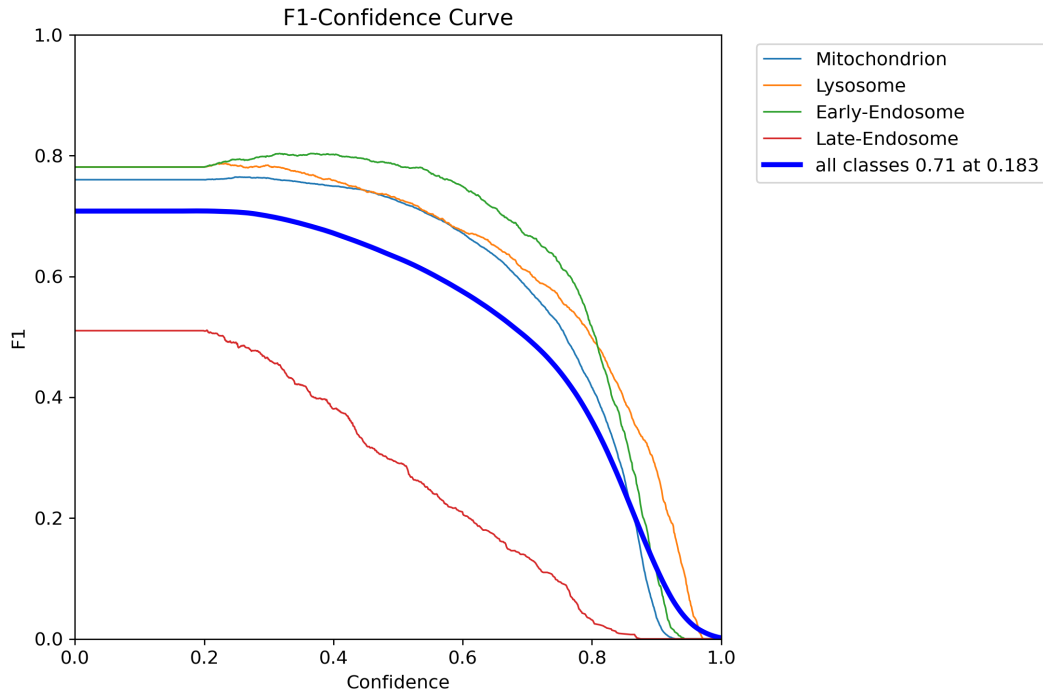


Figure 4.22: F1-Confidence Curve for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

mostly correct with an increased confidence level and since the Precision value never dips below 0,6 the model still predicts well at lower Confidence levels.

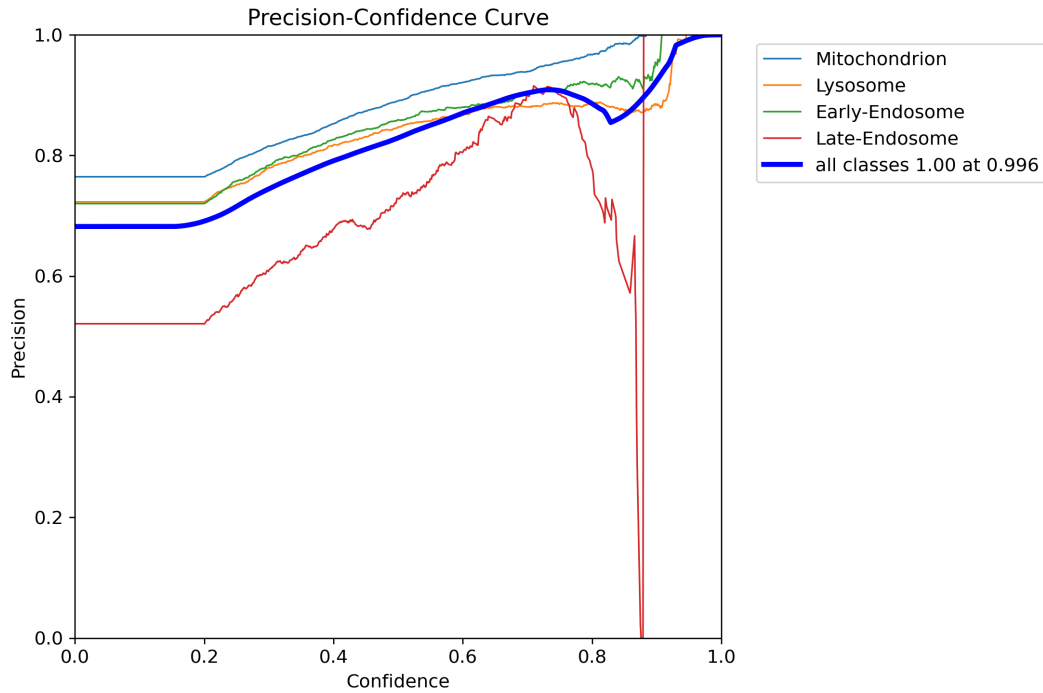


Figure 4.23: Precision-Confidence Curve for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

Figure 4.24 is the Recall-Confidence Curve. It peaks at 0 Confidence level with a Recall value of 0,74, capped due to the chosen Confidence level of 0,2 for the test phase. Like in the validation phase, it gradually dips with the increase of the Confidence level, indicating that the model could be performing more predictions when it's Confidence is low, with some predictions coinciding with the Ground Truth, and failing most True Values when it's more confident.

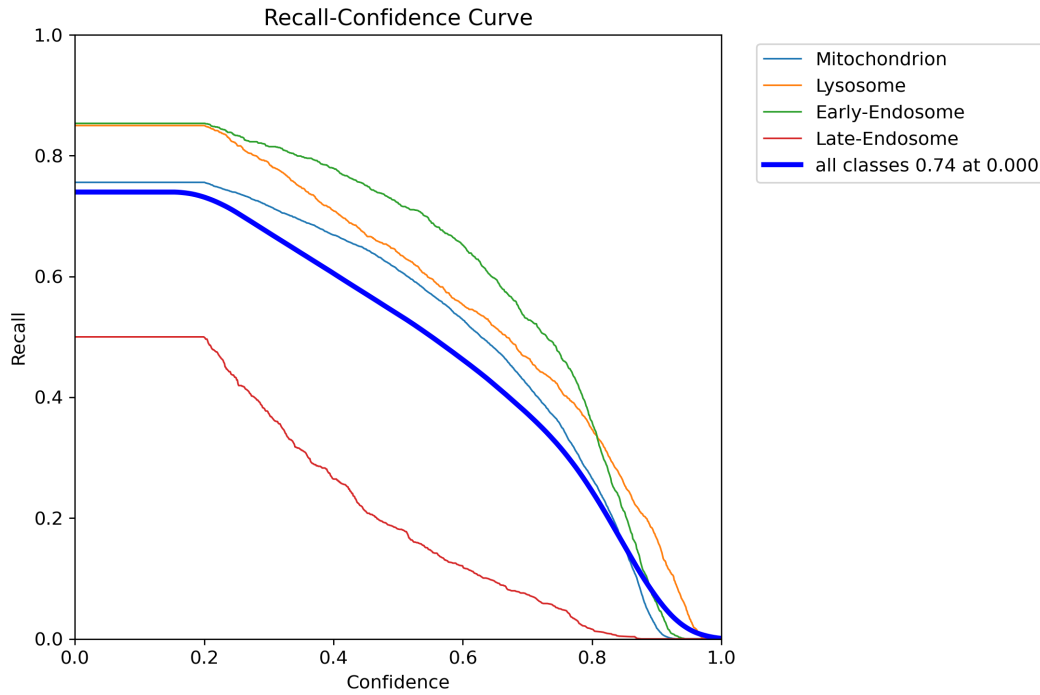


Figure 4.24: Recall-Confidence Curve for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

Figure 4.25 is the Precision-Recall Curve, relating the two metrics. Mapping both metrics at the same Confidence values, it becomes apparent that when the Intersection over Union threshold is set to 50% the mean Average Precision reaches a value of 0,726. This value shows an improvement over the validation phase of the training. This upward shift could mean that the test set is easier than the validation set (less noise, more contrast, larger and/or more visible objects) aiding the model in its task, although this is only intuition.

Figure 4.26 is the Normalised Confusion Matrix. It shows that Late-Endosomes are often classified as other classes or background, once again highlighting the similarity between the classes and the need for class-level augmentation, but also indicating that Mitochondria, Lysosomes and Early-Endosomes are more unique in comparison.

Overall, the results obtained from the training and testing of the YOLOv8 segmentation model demonstrate the model's reliability for the task. Although a confidence level of 20% separated two distinct results the quantitative evaluation indicated values

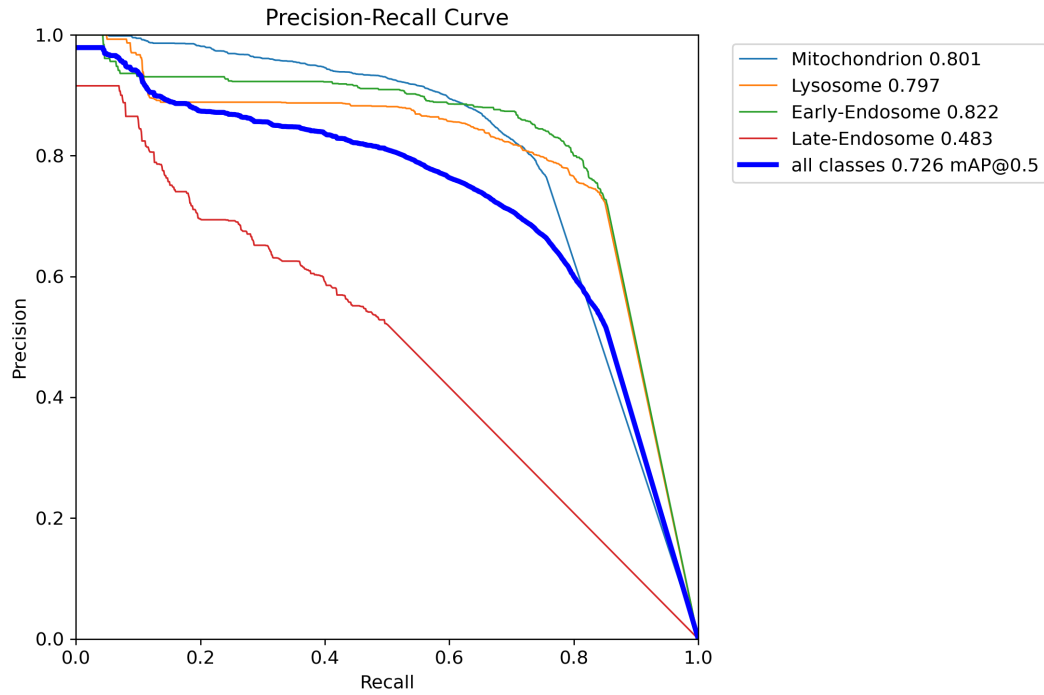


Figure 4.25: Precision-Recall Curve for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

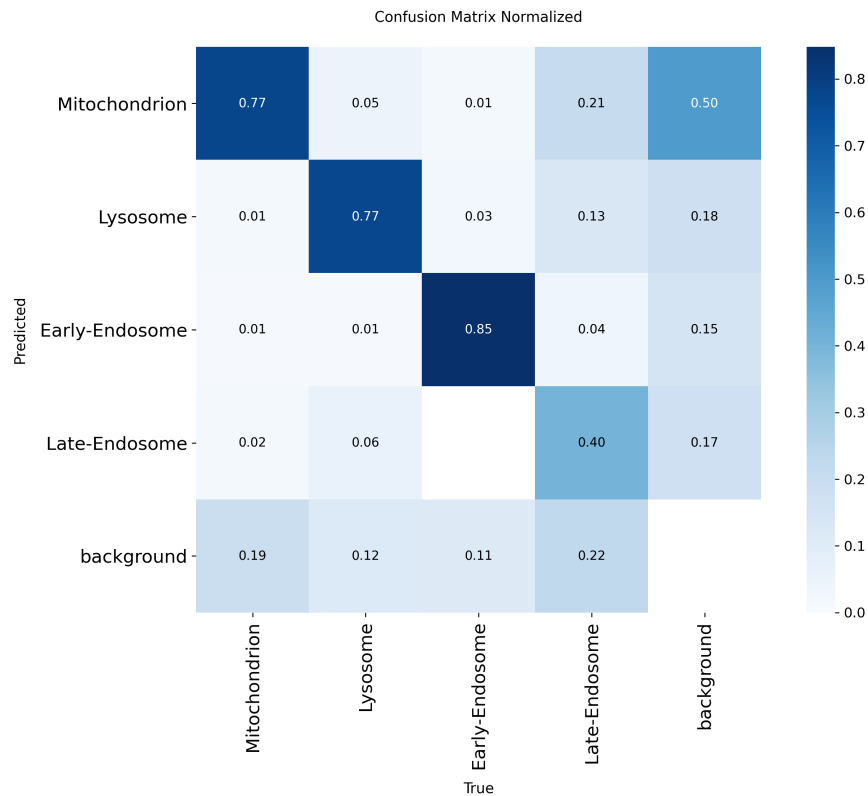


Figure 4.26: Normalised Confusion Matrix for Test for Data Augmentation, Multi-Plane Sampling and Hyper-parameter Tuning

higher than 65% for most test metrics, indicating a high level of accuracy in both detection and segmentation. These metrics suggest that the model makes more predictions with lower confidence and more correct predictions with higher confidence, with the F1-Score finding a common ground for quality and quantity. Qualitative evaluation corroborates this statement with fewer predictions compared with the detection task, with confidence values balanced throughout the segmentation prediction. Together, these metrics point towards a high reliability of YOLOv8 as a segmentation model for low contrast, noisy images.

At the time of writing of this document, Ultralytics released the version YOLO26 of the YOLO model family [45]. A model, YOLO26m-seg.pt, was trained under the same conditions as the model in this section with the same train and validation sets, being tested on the same test set. In order to compare the results with the current state of the art in YOLO segmentation on FIB-SEM images both models will be contrasted class-wise against the model by Nešić et al. [31], who trained a YOLOv8-based model and tested it with two distinct manually acquired ground truths.

Table 4.15: Class-wise Precision Results

Class	Nešić et al. M1 [31]	Nešić et al. M2 [31]	YOLOv8 (Author)	YOLO26 (Author)
Mitochondrion	0,84	0,89	0,765	<u>0,842</u>
Lysosome	0,73	<u>0,74</u>	0,723	0,811
Early-Endosome	0,84	<u>0,82</u>	0,720	<u>0,836</u>
Late-Endosome	0,38	0,18	<u>0,521</u>	0,621
Average	<u>0,698</u>	0,658	<u>0,682</u>	0,778

Table 4.16: Class-wise Recall Results

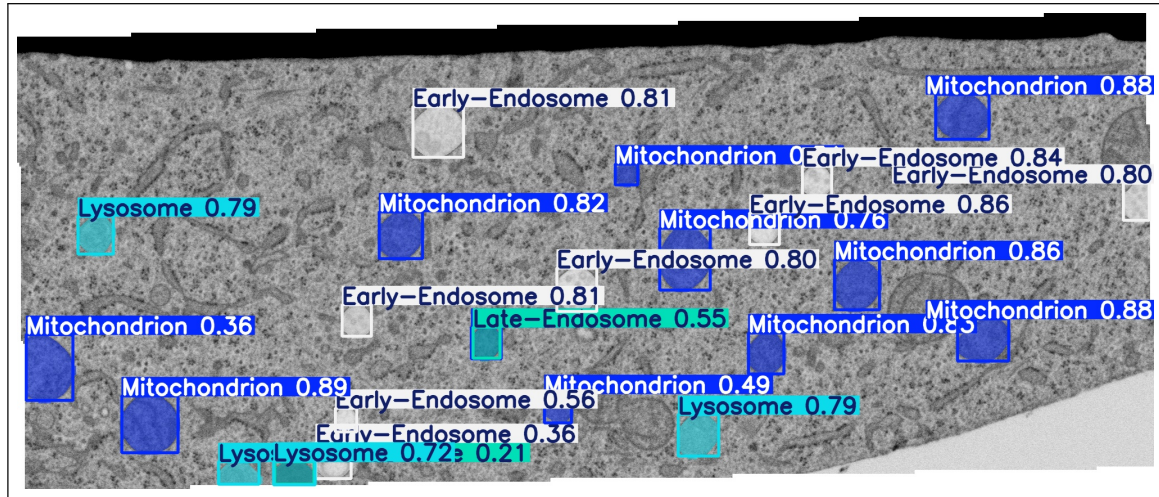
Class	Nešić et al. M1 [31]	Nešić et al. M2 [31]	YOLOv8 (Author)	YOLO26 (Author)
Mitochondrion	<u>0,92</u>	0,93	0,756	0,597
Lysosome	<u>0,83</u>	0,75	0,850	0,702
Early-Endosome	<u>0,80</u>	0,67	0,853	0,764
Late-Endosome	0,14	0,11	0,500	<u>0,318</u>
Average	<u>0,673</u>	0,615	0,740	<u>0,595</u>

Table 4.17: Class-wise F1-Score Results

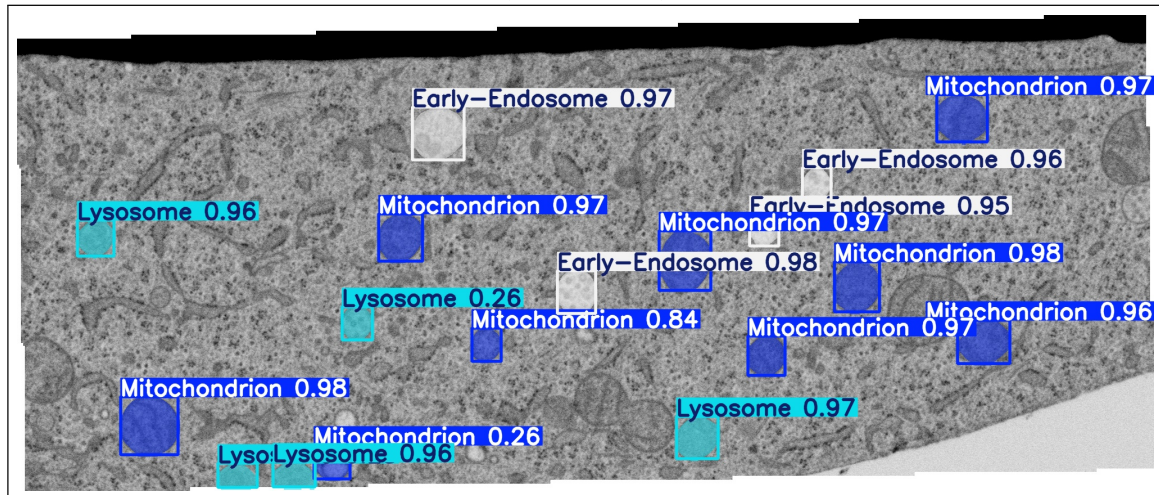
Class	Nešić et al. M1 [31]	Nešić et al. M2 [31]	YOLOv8 (Author)	YOLO26 (Author)
Mitochondrion	0,88	0,91	0,76	0,70
Lysosome	<u>0,78</u>	0,74	0,78	0,75
Early-Endosome	0,82	0,74	0,78	0,80
Late-Endosome	0,20	0,14	0,51	0,42
Average	0,67	0,63	0,71	<u>0,67</u>

Tables 4.15 and 4.16 group the comparison of the four models in regards to Precision and Recall, respectively. Comparing Nešić et al.'s work to the models obtained for this project it is clear that they are in tandem with each other. While their model achieves better performance in every class apart from Late-Endosomes, the trained models for

this project are a bit less achieving but are able to also work with Late-Endosomes. Comparing YOLOv8 to YOLO26 it is noticeable that Precision metrics by YOLO26 are better than YOLOv8, but Recall is better with YOLOv8. Doing a harmonic mean, also known as the F1-Score, to the Precision and Recall results in Figure 4.17. Additionally, a prediction was conducted on the image in Figure 4.21.(a) with both models, displayed in Figure 4.27.



(YOLOv8)



(YOLO26)

Figure 4.27: Segmentation of organelles done by YOLOv8 and YOLO26

5 IMAGE COMPRESSION WITH JPEG AI

Image compression is a method for reducing the size of an image while retaining as much important information as possible and ensuring that the compressed image remains closely similar to the original. In this chapter, the focus is on the JPEG AI standard [26], a learning-based codec that employs deep learning to optimize compression based on human visual perception as well as performance in image processing and computer vision tasks. JPEG AI aims to deliver a coding solution that matches or surpasses existing image compression standards in efficiency, while maintaining comparable visual quality. Furthermore, it is designed to enable efficient processing directly in the compressed domain, facilitating machine learning-based image analysis applications. Figure 2.10 in Section 2.3.3 contains the framework of JPEG AI [26], illustrating how the input image in the encoder codes it to bit stream files to be decoded into reconstructed images.

5.1 Overview of the coding software

The available code package includes a *README* file describing the general execution workflow, required dependencies, and example command-line instructions for running specific tasks. After outlining the necessary software installations and updates, the documentation presents a set of commands intended to execute individual compression tasks with specific inputs. The documentation also provides a command that executes all configurations supported by the codebase, enabling the evaluation of all bit rate settings, tools, and compression methods.

The execution flow of the JPEG AI code can be summarized as follows. First, the images to be compressed are placed in a predefined folder automatically accessed by the system. A script is then executed to run all available JPEG AI configurations. During this process, the code selects an operation point between Base and High: Base uses a neural network structure with simpler operations suitable for low-complexity devices, while High employs more advanced tools to achieve better compression quality at the cost of increased complexity. After this selection, the compression methods are executed.

Only a subset of tools specified in a JavaScript configuration file are enabled for compression, namely ECThread8, ICCI, LSBS, RDLR, and ResVarScale. Different execution modes are defined depending on whether certain tools are enabled or disabled. When tools are disabled, the available execution modes include BRM (Bit Rate Matching),

which aims to match the achieved bit rate to a target value; CPU, where all computations are performed on the CPU; encCPU–decGPU, where encoding runs on the CPU and decoding on the GPU; encGPU–decCPU, where encoding is performed on the GPU and decoding on the CPU; GPU, where all computations are executed on the GPU; and GPU-LH, a GPU-only mode with an additional LH specification, hypothesized to be related to the Discrete Wavelet Transform (DWT) algorithm due to the use of Low-High (LH) 2D-DWT subbands as proposed by Roman Starosolski [54].

When all tools are enabled, only the BRM and GPU execution modes are available.

For each execution mode, the code iterates through a predefined set of bit rates. By default, JPEG AI evaluates five target bit rates corresponding to 6%, 12%, 25%, 50%, and 75% of the original image bit rate. These values are recommended by the developers and defined in a JavaScript array. Although the documentation suggests that users can modify this array to include additional bit rates, attempts to extend it using values from online sources resulted in execution errors (error code 18), leading to the use of the default values.

As previously mentioned, the JPEG AI pipeline requires only the input images for the testing phase. According to the documentation, input images should have bit depths of 8 or 10 bits per pixel, with minor adjustments required for 16-bit images to ensure compatibility. The outputs generated depend on the selected operation point and execution mode.

After execution, the output directory contains two main folders corresponding to the “base” and “high” operation points. Within each of these, subfolders are created for each execution mode. Each execution mode folder contains five subfolders with distinct purposes. The “bit” folder stores binary bitstream files representing compressed images at different bit rates; their internal structure is undocumented and cannot be opened with standard software tools. The “log” folder is divided into three subfolders: “compare”, which stores logs comparing MD5 hashes of reconstructed images; “dec”, which includes decoding logs, MD5 hashes, and decoding-related metrics; and “enc”, which contains encoding logs, MD5 hashes, and encoding-related metrics. The “ori” folder is empty and is assumed to have previously stored copies of the input images, likely removed to avoid redundancy and excessive storage usage. The “rec” folder contains images presumed to be reconstructed directly from the original images without using the bitstreams. Finally, the “rec_dec” folder contains images reconstructed by decoding the bitstreams, which are expected to be compared with the “rec” images using MD5 hash comparisons stored in the log/compare directory.

At the end of execution, results from all modes are compiled into TXT files located within the operation point folder, providing a general overview of the performed steps. Additionally, CSV files are generated containing metric values for each operation point, execution mode, and target bit rate.

5.2 Quantitative evaluation metrics

As a Deep-Learning based model, JPEG AI relies on certain metrics comparing the original image to its compressed counterpart and help determine the effectiveness of the trained model. Some metrics will produce data regarding the machines vision, others will emulate the human eye and how it perceives images on a computer, and others will evaluate the balance between noise and useful data on the images. As stated in the previous section, each image in the test set was compressed for one of two operation points, one of six compression methods and one of five target bit rates, and each of the images will produce the metrics introduced in this section.

Multi-Scale Structural Similarity Index Metric [55], or MS-SSIM, is used to compute luminance, contrast and structure at the same time over several scales/resolutions. The formula is expressed as:

$$\text{MS-SSIM}(x, y) = [l_M(x, y)]^{\alpha_M} \prod_{j=1}^M [c_j(x, y)]^{\beta_j} [s_j(x, y)]^{\gamma_j}$$

where x and y are the reference and distorted images, M is the number of scales, l_M is the luminance comparison at the coarsest scale c_j and s_j are the contrast and structure comparisons at scale j , and α_M , β_j and γ_j are weighting exponents. Its value varies from 0 to 1, with the maximum being obtained when $x=y$, typically indicating lossless compression.

Peak Signal-to-Noise Ratio [56], or PSNR, measures, in decibels, the ratio between the max pixel value and the noise, expressed through the Mean Square Error, between the two analysed images. It is expressed as:

$$\text{PSNR} = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right)$$

with MAX translating to the max value of a pixel and MSE to the aforementioned Mean Square Error with:

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (I(i, j) - K(i, j))^2$$

where M and N are the dimensions of the image, I is the reference image and K is the reconstructed image. Typical values are between 30 dB and 50 dB, where a higher value implies a smaller MSE, with values above 40 dB considered very good and below 20 dB discarding the images altogether. This metric was applied to the YUV equivalent to RGB images, isolating the information from the colors.

Visual Information Fidelity [57], or VIF, is based on information theory and is typically

defined as the ratio of mutual information between the reference and distorted images, relative to the reference and the human visual system (HVS). It is expressed as:

$$\text{VIF} = \frac{\sum_{i=1}^N I(\mathbf{C}_i; \mathbf{F}_i | \mathbf{s}_i)}{\sum_{i=1}^N I(\mathbf{C}_i; \mathbf{E}_i | \mathbf{s}_i)}$$

where I is mutual information, C_i are the wavelet coefficients of the reference image, F_i are the wavelet coefficients of the distorted image, E_i are the wavelet coefficients of the reference image passed through the HVS model, s_i represents the local variance, i indexes subbands/scales and N is the number of subbands. It possesses values from 0 to 1, with 1 standing as high similarity between reference and distorted images.

Feature Similarity Index Metric [58], or FSIM, is a full-reference image quality assessment metric based on the hypothesis that the Human Visual System (HVS) primarily perceives images through low-level features, particularly phase congruency and gradient magnitude. Phase congruency captures perceptually significant structures such as edges and corners, while gradient magnitude represents local contrast information.

FSIM is computed as a weighted average of local similarity measures over the image domain and is expressed as:

$$\text{FSIM} = \frac{\sum_{\mathbf{x} \in \Omega} S_L(\mathbf{x}) \cdot PC_m(\mathbf{x})}{\sum_{\mathbf{x} \in \Omega} PC_m(\mathbf{x})}$$

where Ω denotes the set of all pixel locations, $PC_m(\mathbf{x}) = \max(PC_x(\mathbf{x}), PC_y(\mathbf{x}))$ is the maximum phase congruency between the reference and distorted images, and $S_L(\mathbf{x})$ is the local similarity measure defined as:

$$S_L(\mathbf{x}) = S_{PC}(\mathbf{x}) \cdot S_{GM}(\mathbf{x})$$

with:

$$S_{PC}(\mathbf{x}) = \frac{2PC_x(\mathbf{x})PC_y(\mathbf{x}) + T_1}{PC_x^2(\mathbf{x}) + PC_y^2(\mathbf{x}) + T_1}$$

$$S_{GM}(\mathbf{x}) = \frac{2GM_x(\mathbf{x})GM_y(\mathbf{x}) + T_2}{GM_x^2(\mathbf{x}) + GM_y^2(\mathbf{x}) + T_2}$$

where PC_x and PC_y denote the phase congruency maps of the reference and distorted images, GM_x and GM_y their gradient magnitude maps, and T_1 and T_2 are small constants introduced for numerical stability.

FSIM values range from 0 to 1, with a value of 1 indicating identical images. Due to its strong correlation with perceptual quality, FSIM is particularly effective at capturing visually relevant degradations.

Normalized Laplacian Pyramid Distance [59], or NLPD, is a perceptually motivated full-reference metric that models early stages of the Human Visual System, meaning

it evaluates both reference and distorted images. It measures image distortion as the distance between normalized Laplacian pyramid representations of the reference and distorted images.

NLPD is defined as:

$$\text{NLPD}(x, y) = \left(\sum_{k=1}^K \frac{1}{N_k} \left\| \frac{L_k^x}{\sigma_k^x + \epsilon} - \frac{L_k^y}{\sigma_k^y + \epsilon} \right\|_2^2 \right)^{1/2}$$

where L_k^x and L_k^y are the Laplacian pyramid coefficients of the reference and distorted images at scale k , σ_k^x and σ_k^y represent local normalization factors modeling contrast gain control, N_k is the number of coefficients at scale k , K is the total number of pyramid levels, and ϵ is a small constant added for numerical stability.

NLPD is a distance metric with a minimum value of zero, achieved when the two images are identical. It has no fixed upper bound, with larger values indicating increasing perceptual distortion.

Information-Weighted Structural Similarity Index Metric [60], or IW-SSIM, is an extension of the Structural Similarity Index that incorporates information content weighting to emphasize perceptually important regions. The underlying principle is that distortions occurring in visually informative areas have a greater perceptual impact than those in less informative regions. IW-SSIM is computed as:

$$\text{IW-SSIM} = \frac{\sum_{j=1}^M w_j \cdot \text{SSIM}_j}{\sum_{j=1}^M w_j}$$

where SSIM_j is the SSIM value computed at scale j , M is the number of scales, and w_j represents an information-based weight derived from the reference image at that scale.

The weights are proportional to the local information content:

$$w_j \propto I_j$$

where I_j is commonly estimated using local variance or entropy measures.

IW-SSIM values range from 0 to 1, with higher values indicating greater structural similarity. Compared to SSIM and MS-SSIM, IW-SSIM exhibits improved correlation with subjective image quality by prioritizing distortions in perceptually significant regions.

Video Multi-Method Assessment Fusion [61], or VMAF, is a perceptual quality assessment metric based on machine learning techniques. Although originally developed for video quality evaluation, it is frequently applied on a frame-by-frame basis for image quality assessment. VMAF combines multiple elementary quality features into a single score using Support Vector Regression (SVR).

The metric is defined as:

$$\text{VMAF} = f(\text{VIF}, \text{DLM}, \text{ANSNR}, \dots)$$

where $f(\cdot)$ denotes a trained regression model, VIF is the Visual Information Fidelity metric, DLM is the Detail Loss Metric, ANSNR is an adaptive noise-weighted PSNR variant, and additional perceptual features may be included depending on the model configuration.

VMAF produces scores typically ranging from 0 to 100, where higher values correspond to better perceptual quality. Values above 90 are generally associated with visually lossless quality, while values below 50 indicate severe degradation.

Peak Signal-to-Noise Ratio with Human Visual System modeling [62], or PSNR-HVS, is a perceptually enhanced variant of PSNR that incorporates characteristics of the Human Visual System, particularly contrast sensitivity and frequency masking effects.

PSNR-HVS is expressed as:

$$\text{PSNR-HVS} = 10 \log_{10} \left(\frac{MAX^2}{\text{MSE}_{\text{HVS}}} \right)$$

where the perceptually weighted Mean Square Error is given by:

$$\text{MSE}_{\text{HVS}} = \frac{1}{MN} \sum_{i,j} w_{i,j} (D_{i,j}^x - D_{i,j}^y)^2$$

with $D_{i,j}^x$ and $D_{i,j}^y$ representing the Discrete Cosine Transform (DCT) coefficients of the reference and distorted images, respectively, and $w_{i,j}$ denoting frequency-dependent weighting factors derived from the contrast sensitivity function of the HVS. Typical PSNR-HVS values are comparable to classical PSNR, usually ranging between 30 dB and 50 dB, with higher values indicating better perceived quality.

Table 5.1 summarizes all the metrics on this section by uniting the minimum and maximum values.

Table 5.1: Evaluated compression metrics and limiting values

Metric	Minimum Value	Maximum Value
MS-SSIM	0	1 (best)
PSNR	30-	50+ (better)
VIF	0	1 (better)
FSIM	0	1 (best)
NLPD	0 (best)	0+
IW-SSIM	0	1 (best)
VMAF	0	100 (best)
PSNR-HVS	30-	50+ (better)

5.3 Preparation of test images

The images to be inputted into the code must adhere to certain rules for the code to recognise them: they can either have 8 bpp or 10 bpp, they must possess three RGB channels, they must be PNG images, their name had to be written in the form “<id(5digits)>_<task>_<width>x<height>.png”, with “id” being a number unique to the image as not to mix them, “task” being for Training, writing TR, Testing, writing TE or Validation, writing VL, “width” and “height” being the dimensions of the image. The images were previously set to RGB images and 8 bits per pixel in preparation for YOLO segmentation, which also operates with PNG images, meaning the remaining preparation would be to rename them. Without scrambling the order of the images, they were renamed to their appropriate name.

5.4 Compression results

After running the pre-trained model on the test set, it became evident that, for both operation points, only six compression modes completed successfully without generating error logs. These modes, listed in the order presented in the results files, are tools_off-GPU, tools_off-GPU-LH, only-ResVarScale, only-ICCI, only-LSBS, and only-ECThread8.

The test images were compressed across all available operation points, compression methods, and target bit rates, and subsequently evaluated using the metrics described in the previous section. The resulting metric values per image were extracted and then averaged across bit rates using an arithmetic mean. This approach was adopted due to the similarity among the images and the assumption that their metric values would not differ significantly. To validate this assumption, the standard deviation was also calculated. The average and standard deviation were computed using Excel’s AVERAGE and STDEV functions, respectively.

Additionally, test image number 50 was selected for visual analysis, as it represents a central element in the dataset while avoiding extreme cases. This image is presented at both the lowest and highest target bit rates to provide a qualitative reference that complements the quantitative metrics.

However, considering that there are two operation points and six compression methods, the analysis results in twelve tables of arithmetic averages, twelve tables of standard deviations, and twenty-four images representing the lowest and highest compression rates. Due to the volume of data, only one compression method for a single operation point will be discussed in detail in this section, while the remaining tables and images are provided in Annexes B through L.

Table 5.2 presents the average values of the extracted metrics for different compressed bitrates from 6% to 75%. It can be observed that MS-SSIM was computed using both Torch [63], a deep learning library for Python, and the Image Quality Assessment [64] (IQA) framework, which evaluates the similarity between reconstructed and original images and is responsible for most of the metrics used in this section. This dual computation may indicate an attempt to validate the results obtained through Torch against a direct implementation of the metric, although this remains speculative.

Even at a 6% bit rate, MS-SSIM values exceed 90%, suggesting that the reconstructed images maintain good perceptual quality, albeit with some visible artifacts. Regarding luminance in the YUV color space, the Peak Signal-to-Noise Ratio ranges from approximately 27 dB to over 30 dB, indicating that luminance—which carries the majority of image information—exhibits more reconstruction errors. In contrast, the chrominance components (U and V) achieve higher PSNR values, even at lower bit rates.

VIF values, which emphasize information fidelity, indicate that higher compression levels result in significant information loss, implying degradation of edges and potential distortion or loss of important structures. However, high FSIM values suggest that perceptually significant features are largely preserved, meaning that the overall structure of the image remains visually intact. Similarly, NLPD values indicate a high perceptual similarity between distorted and reference images.

IW-SSIM, another information-based metric, consistently exceeds 90% across all evaluated bit rates, further supporting the conclusion that major structures are preserved while finer details are lost. VMAF provides a more detailed distinction across compression levels, showing noticeable degradation at lower bit rates and only minor quality loss at higher bit rates. Finally, PSNR values adjusted to simulate the Human Visual System align with VMAF results, where lower values correspond to greater visual degradation and values around 35 dB indicate images that appear similar to the original, with only minor differences.

Table 5.2: Average values of metrics for Base operation point on tools off - GPU compression

Target Bit Rates	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75%	0,99	0,99	31,51	54,43	56,19	0,51	0,99	0,13	0,98	88,07	35,49
50%	0,98	0,98	30,84	57,08	56,77	0,47	0,99	0,15	0,98	85,70	33,17
25%	0,97	0,97	28,91	57,16	57,98	0,39	0,99	0,20	0,97	75,12	29,34
12%	0,96	0,96	27,92	56,01	56,87	0,35	0,98	0,24	0,95	66,44	27,03
6%	0,93	0,92	26,55	56,65	57,35	0,28	0,97	0,32	0,91	46,43	23,82

Table 5.3 contains the Standard Deviation values of the metrics generated by JPEG AI during the test session. Their low values across all metrics and compression levels solidify the claim that since the images were extracted from the same volume, all have low contrast and are noisy, they are similar enough that the calculated metrics don't differ much from each other.

Table 5.3: Standard Deviation values of metrics for Base operation point on tools off - GPU compression

Target Bit Rates	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75%	1,5e-4	1,5e-4	1,1e-1	1,2e-1	2e-1	1,8e-3	6e-5	8,2e-4	6,5e-5	1,1e-1	9,6e-2
50%	2,7e-4	2,8e-4	1,1e-1	1,2e-1	1,6e-1	2,1e-3	4,6e-5	8,4e-4	1,4e-4	1,2e-1	8,7e-2
25%	2,8e-4	3,1e-4	1,3e-1	3,4e-1	2,5e-1	1,9e-3	9,9e-5	1,7e-3	2,2e-4	2,8e-1	1e-1
12%	3,7e-4	3,9e-4	1,4e-1	2,3e-1	3,7e-1	1,8e-3	1,3e-4	2,4e-3	4,6e-4	3,3e-1	1,1e-1
6%	1,1e-3	1e-3	1,4e-1	6,3e-1	4,8e-1	1,5e-3	4e-4	3,5e-3	1,5e-3	4,6e-1	1,2e-1

From a quality assessment perspective, Figures 5.1 and 5.2 present two reconstructed images for this specific combination of operation point and compression method, corresponding to the lowest and highest allowed bit rates, respectively. Although the differences are subtle, certain structures, such as the boundaries between organelles and the background, appear slightly blurred or less distinct at lower bit rates. This observation is consistent with the results indicated by visually and information-oriented metrics such as VIF, IW-SSIM, VMAF, and PSNR-HVS.

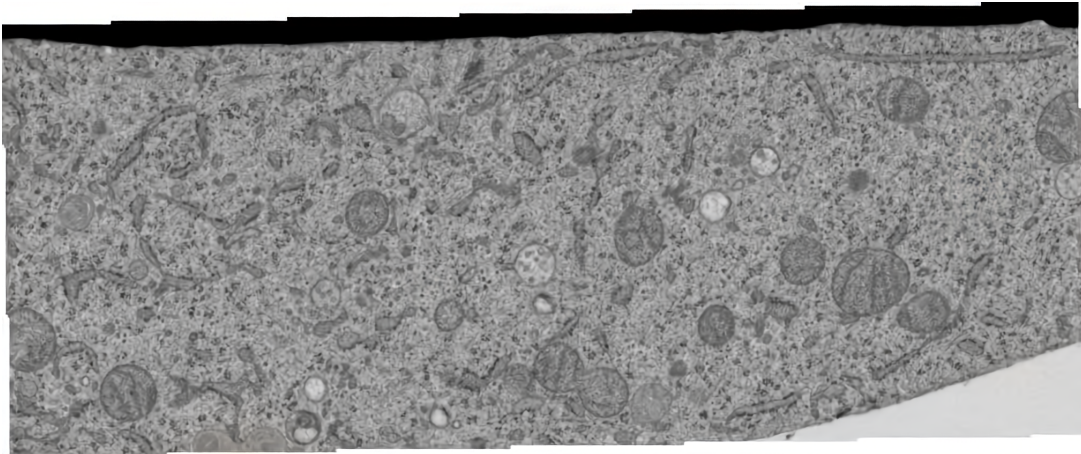


Figure 5.1: Reconstructed image of Base operation point on tools off - GPU compression at 6% bit rate

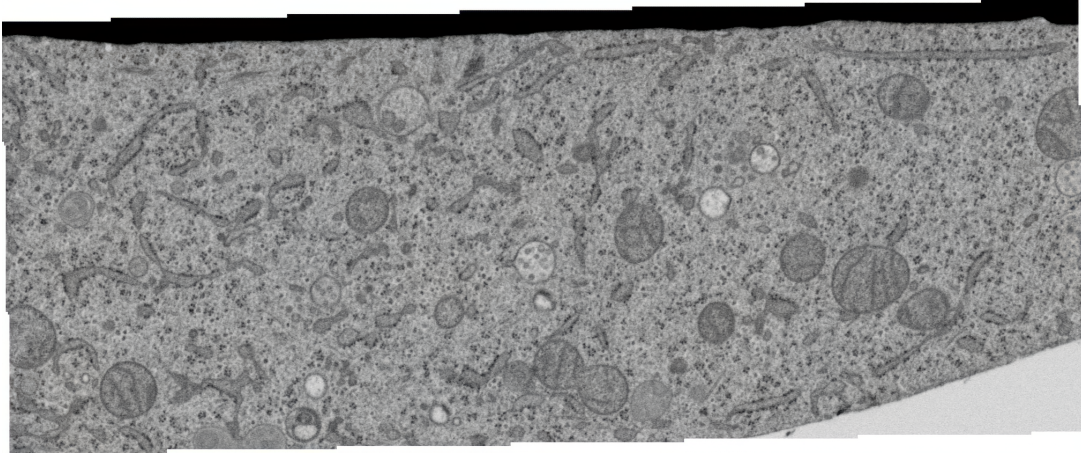


Figure 5.2: Reconstructed image of Base operation point on tools off - GPU compression at 75% bit rate

Overall, the results indicate that the pre-trained JPEG AI model achieves high fidelity according to traditional distortion-based metrics, such as PSNR and MS-SSIM, particularly when considering the individual color channels in the YUV representation. These metrics suggest that, from both pixel-wise accuracy and structural similarity perspectives, the reconstructed images remain close to the original, even at relatively high compression levels.

However, the consistently lower scores observed in vision-based metrics like VIF, NLPD, VMAF and PSNR HVS reveal a different trend: perceptual quality degrades as the bit rate decreases, with compression artifacts and visual distortions becoming increasingly noticeable, especially in the range from 50% down to 6% bit rate. This discrepancy highlights the limitations of relying exclusively on conventional metrics to evaluate perceptual quality, particularly when the model has not been specifically trained for the target domain.

These findings suggest that, although JPEG AI effectively preserves numerical similarity, perceptual degradation may still occur. This reinforces the importance of using complementary evaluation metrics and, potentially, adapting or fine-tuning models for domain-specific applications.

6 COMPRESSED IMAGE SEGMENTATION

This section outlines the followed methodology to infer a previously trained YOLOv8 model with images compressed using JPEG AI. Building upon the earlier stages of dataset preparation, model training, and compression analysis, the focus here is to define a structured plan for evaluating how advanced learned compression affects downstream object detection performance.

As explained in Chapter 4, the available dataset was split into three sets for YOLOv8 manipulation, train, validation and test. The model was trained using the training set, consisted of Multi-Plane images from the FIB-SEM volume the images were extracted from, and performed the val task using the validation set. The test set consists of an array of data relating to the later half of the original, unaltered images of the dataset.

In Chapter 5, JPEG AI was used to compress the images. Due to limitations in the available code, training was not a possibility, with the test being performed on a pre-trained version. Analysing the metrics, it was apparent that the alterations suffered by the images were not severe enough that they would degrade them beyond not being useful even at the lowest target bit rate, maintaining high similarity and Human Vision System metrics.

The approach in this chapter aims to ensure consistency and fairness in the evaluation by maintaining identical inference conditions across original and compressed images, isolating the impact of JPEG AI compression. By applying different compression levels and observing how they affect detection and segmentation behavior, this approach helps connect model development with quantitative analysis.

Within each operation point and each compression method the reconstructed images were grouped together. This means that across the 2 operation points and 6 compression modes all images were in one folder, difficulting automatic access and manipulation. With 656 input JPEG AI images with an average of 3,3 megabytes of space usage, the total memory occupation was 26 Gigabytes of reconstructed images alone, increasing to over 200 Gigabytes when accounting for other output files. To prevent an increase in memory usage, for each time the model would test on a given bit rate percentage it would copy only the needed data to a designated folder where it could access the files with the locations of the organelles, emptying the folder after each run before looping the operation points, compression modes and target bit rates.

Since the model used was trained with square images of dimensions 512 by 512 pixels, the same algorithm would be applied to the compressed set for the validation portion

of the testing, utilizing a full image for the prediction. The model would be configured to validate on the test set with the compressed images. The main arguments in play were *imgsz* with a value of 512, *conf* with a confidence threshold of 0,2 ensuring the model would only predict if the confidence level was 20% or greater, *iou* with an Intersection over Union threshold of 0,5 to only consider predictions where the Ground Truth covered over 50% of the prediction, and *split* set to "test" indicating that the function "val" would utilize the test set instead of the validation set. This function outputs the metrics used to evaluate how the compression levels affected the models ability do identify the organelles. Additionally, the YOLO function "predict" would predict and display segmentation mask and bounding box of the predictions with the confidence level of the prediction on the 50th image of each test set, allowing for a qualitative analysis of the behaviour of the model.

Results

Analysing the base operation point with tools off GPU compression mode, as was the evaluated case in Chapter 5, it is possible to analyse the metrics outputted by the application of the trained YOLOv8 model on the compressed images. Table 6.1 contains the mask relative Precision, Recall, mean Average Precision across 50% to 95% Intersection over Union thresholds and the F1-Score contrasting the uncompressed images and the compression stages of the base operation point on tools off GPU compression mode.

Table 6.1: Quantitative results from applying the segmentation model on base operation point on tools off GPU compression mode

Target Bit Rates	Precision	Recall	mAP@50	mAP@50-95	F1-Score
Uncompressed	0,6822	0,7398	0,7257	0,5319	0,7098
75%	0,6834	0,7066	0,7220	0,5230	0,6948
50%	0,7036	0,6870	0,7156	0,5142	0,6952
25%	0,6414	0,6926	0,6874	0,5020	0,6660
12%	0,6152	0,6305	0,6309	0,4621	0,6228
6%	0,5406	0,4870	0,5029	0,3663	0,5124

Initially, the focus of the table resides in the high values. Precision, Recall and, as such, F1-Score, were able to achieve mostly above average values before a general drop in 12% and 6% target bit rates. Plotting the metrics aids with a visual analysis.

The plot in Figure 6.1 provides a visual aid to the behaviour of the segmentation model across compression rates. The metrics values remain mostly non-varying from the uncompressed set to 25% target bit rate, before a decline to 12% and a more accentuated decline to 6%.

Performing a class-wise evaluation on the segmentation metrics across the compression stages of Base operation point and tools-off GPU compression mode provides more

Cell Segmentation in FIB-SEM Microscopy Images

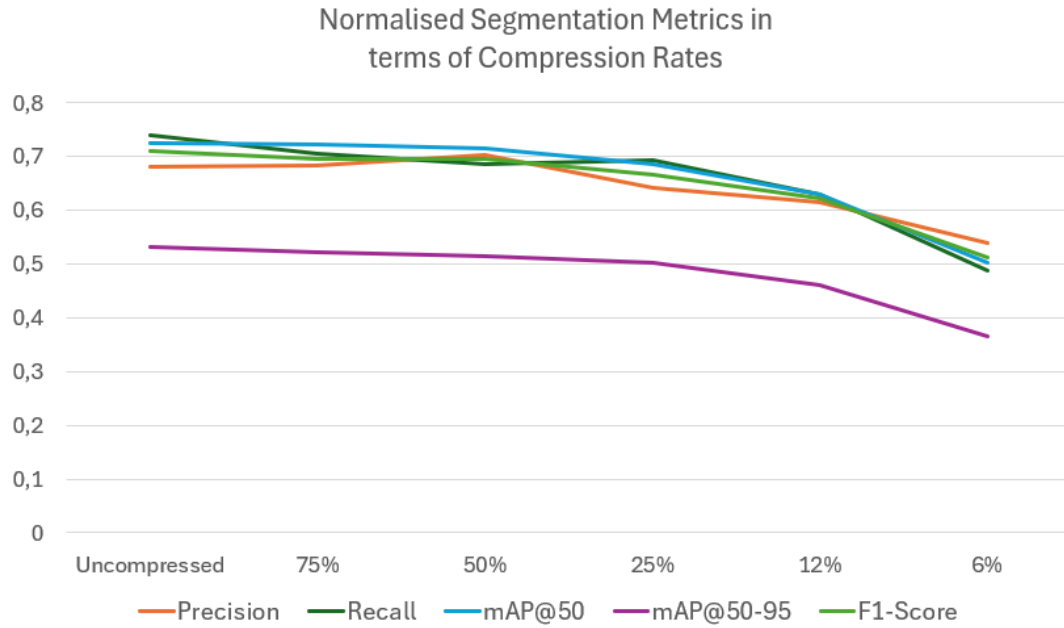


Figure 6.1: Evolution of Segmentation Metrics means for all classes with Compression Rates

insight on the segmentation-compression relationship. Figure 6.2 showcases the four classes' segmentation metrics with compression rates.

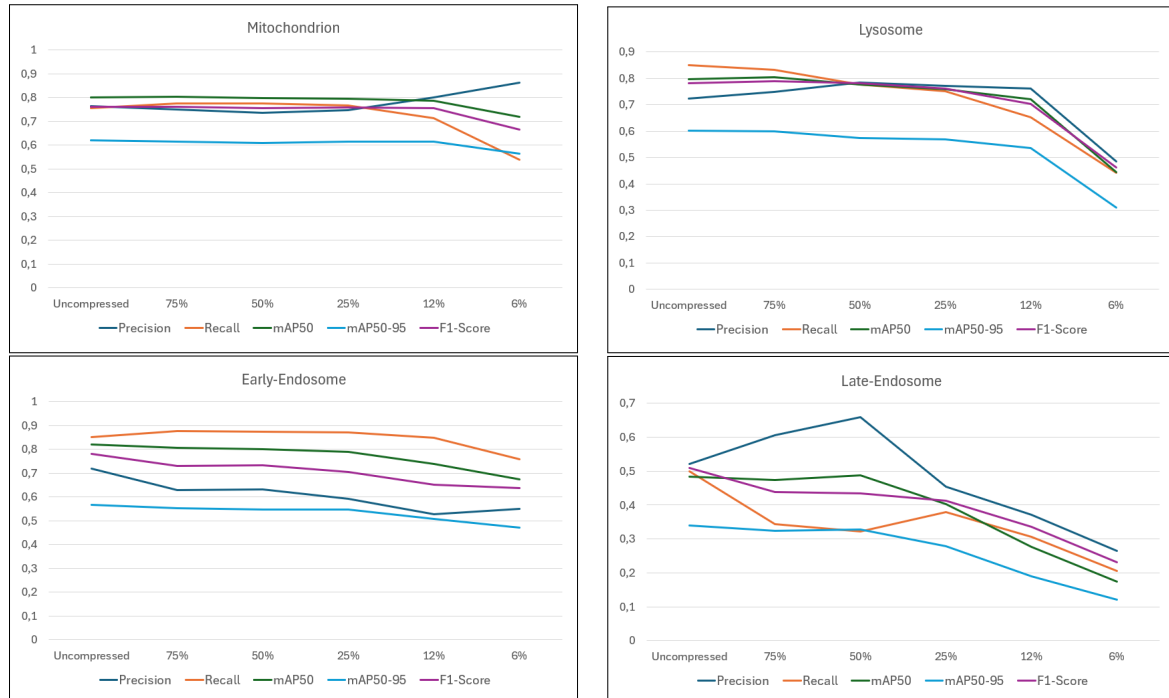


Figure 6.2: Segmentation metrics of each class according to compression stage

Mitochondria segmentation performance was lightly affected by compression, implying that the characteristics of the class are unique enough that content removed by light compression doesn't impede or alter the model's results, until a spike in Precision and

a dip in the other metrics which suggest more of the predictions are correct but less of the ground truth is being predicted on due to the advanced compression rate.

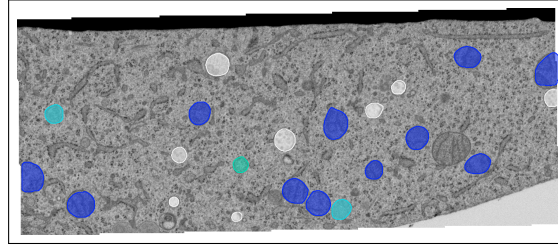
Lysosomes' plot is similar to Figure 6.1 with a slow descent in segmentation metrics according to compression rate. Like in Mitochondria segmentation, Precision increases with compression sooner, indicating that removal of non-key data is affecting the Lysosome by improving the True Positive/Prediction ratio.

Early-Endosomes' Recall metric increases with low compression rate unlike the other plots, meaning more True Positives are being considered as compression increases, lowering at high compression with the increase in Precision.

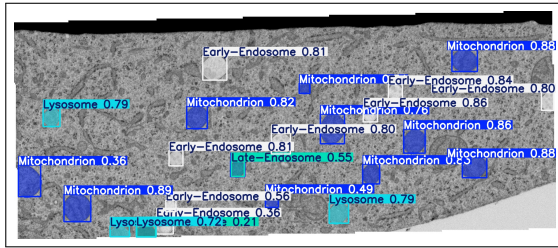
Late-Endosomes, the less common class, contains low metrics throughout compression, with Precision rapidly increasing with low compression rates and Recall is reduced substantially, meaning that True Positives are also reduced but less predictions are being made with a confidence level above the threshold of 0,2 resulting in lower False Negatives, and at 25% compression rate Precision decreases and Recall increases, implying that True Positives are being considered justifying the Recall and more predictions being made which increase the False Negatives, lowering Precision.

From a qualitative perspective, Figure 6.3 presents the predictions made by the model with Mitochondria in blue, Lysosomes in cyan, Early-Endosomes in white, and Late-Endosomes in green. It can be seen that even with finer detail failing to be reconstructed in the representation of the compressed images, the segmentation model is still able to maintain good performance. The occasional failure to correctly or confidently detect/segment one or more organelles reflects the results in Table 6.1 and Figure 6.1, with a smooth descent into unconfident and incorrect predictions.

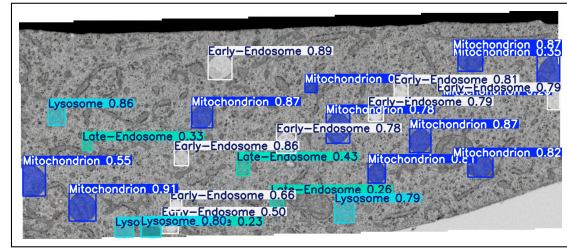
Cell Segmentation in FIB-SEM Microscopy Images



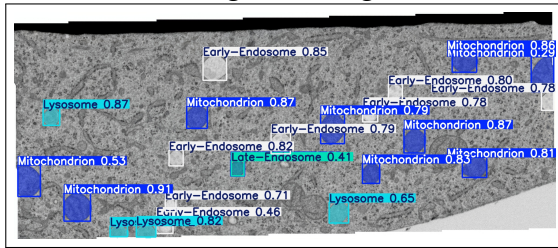
Position of Organelles on the Image



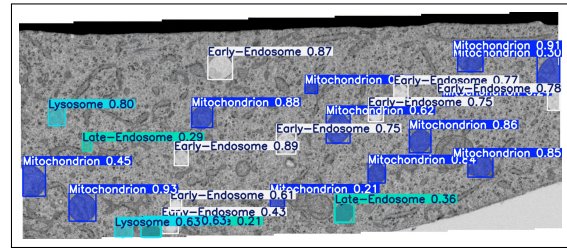
Original Image



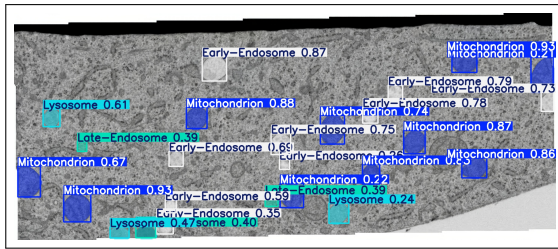
75% Bit Rate



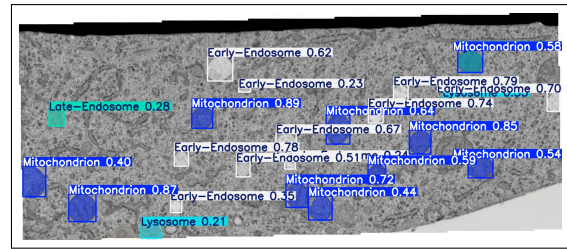
50% Bit Rate



25% Bit Rate



12% Bit Rate



6% Bit Rate

Figure 6.3: Predictions of the yolov8m segmentation model on compressed Images

7 CONCLUSION

This project investigated the impact of image compression on the performance of deep learning-based segmentation models applied to FIB-SEM images. The study focused on the training and evaluation of YOLOv8 models under different experimental setups, with particular attention to dataset handling, training methodology, and inference configuration.

Initially, a detection model was developed to assess if YOLOv8 was capable of detecting the considered organelles within the images before proceeding with the segmentation. In the early experiments, although following standard practices such as random dataset splitting, due to the specific data characteristics, that led to sequential image frames being spread across the training, validation, and test sets. While this setup led to good performance, it did not reflect a realistic evaluation scenario and risked data leakage. Recognizing this limitation, a final model was trained following a more careful experimental protocol, ensuring strict separation between dataset components, exploring the data's volume characteristic to use more data, and providing a more reliable assessment of generalization performance.

To evaluate the effect of compression, the images in the test set were compressed using a pre-trained JPEG AI model. The compression process achieved high compression ratios and favorable objective compression metrics, demonstrating the effectiveness of the method in significantly reducing data size while maintaining overall visual quality. This result highlights the potential of learned compression techniques for storage and transmission of high-resolution FIB-SEM data.

The YOLOv8 model demonstrated strong robustness when applied to compressed images, maintaining high performance metrics across the evaluated target bit rates, from the original uncompressed test set, up to 6% of the original images' raw bitrate. Although a gradual decline in performance was observed as compression increased, the degradation occurred smoothly rather than abruptly, indicating that the model retained its ability to detect relevant features even under significant compression. A class-wise observation demonstrated the differences each class possessed on the uncompressed state and with each compression rate through the segmentation metrics provided by the segmentation model's validation.

Overall, these results highlight the potential of combining modern object detection models with efficient image compression in practical scenarios where storage and transmission constraints are relevant. The demonstrated robustness of YOLOv8 suggests

that meaningful reductions in image bit rate can be achieved without severely compromising detection performance. This is particularly important for applications involving large-scale image datasets or bandwidth-limited environments, where compression is often unavoidable. Consequently, the findings support the usage of deploying deep learning-based detection systems on compressed imagery while maintaining reliable performance, encouraging further exploration of compression-aware pipelines and optimization strategies for real-world medical and visual analysis applications.

The conclusions of this study are restricted by several limitations, such as:

- The dataset consisted of images from a single scan, constraining the data split for testing the developed segmentation model, and consequently impacting its effective generalization capability assessment;
- Annotating microscopic cells requires high technical expertise, and even trained professionals might disagree from each other at some instances, as recognized by N. Nešić et al. [31]. Under a supervised learning setting, missing labels can lead to missing predictions, which reflects in lower recall;
- The compression was not severe for low bit rates, as objects can still be identified easily (see Figures 5.1 and 5.2);
- Segmentation metrics were normalised, not explicitly providing information on number of predictions and correct predictions across confidence levels;
- Even though more recent versions of Ultralytics were explored under a small grid of hyper-parameters, one can never be sure that the optimal values for the hyper-parameters have been found.

Future work should involve:

- Obtaining more data;
- Improve annotations/ensuring a correct ground truth;
- Experiments with severe compression;
- Training of a Deep Learning-Based Compression Model like JPEG AI;
- Design custom methods to evaluate the impact of Gaussian noise in images as well as the confidence threshold and IoU threshold;
- Better hyper-parameter tuning to find the optimal training conditions.

BIBLIOGRAPHIC REFERENCES

- [1] A. M. Mustafa, “Magnetic resonance imaging,” Rush University Medical Center, Tech. Rep., Nov. 2023.
- [2] —, “Computerized tomography,” Rush University Medical Center, Tech. Rep., Nov. 2023.
- [3] G. Prieto, “Fib-sem tomography,” in *Encyclopedia of Membranes*, E. Drioli and L. Giorno, Eds. Berlin, Germany: Springer, 2016, pp. 770–772.
- [4] C. Kizilyaprak, J. Daraspe, and B. M. Humbel, “Focused ion beam scanning electron microscopy in biology,” *Journal of Microscopy*, vol. 254, no. 3, pp. 109–114, 2014. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/jmi.12127>
- [5] L. Zhao, Y. Cui, J. Li, Y. Xie, W. Li, and J. Zhang, “The 3d controllable fabrication of nanomaterials with fib-sem synchronization technology,” *Nanomaterials*, vol. 13, no. 12, p. 1839, 2023. [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10303019/>
- [6] M. W. Phaneuf, *FIB for materials science applications – a review*. Boston, MA, USA: Springer US, 2005, pp. 143–172.
- [7] K. L. Briggman and D. D. Bock, “Volume electron microscopy for neuronal circuit reconstruction,” *Current Opinion in Neurobiology*, vol. 22, no. 1, pp. 154–161, 2012.
- [8] G. Knott, H. Marchman, D. Wall, and B. Lich, “Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling,” *Journal of Neuroscience*, vol. 28, no. 12, pp. 2959–2964, 2008.
- [9] D. H. Hall, E. Hartweg, and K. Nguyen, “Focused ion beam (fib) scanning electron microscopy (sem),” *WormAtlas*, Jun. 2024.
- [10] Y. Liu, J.-m. Xue, and H. Li, “The study on the image thresholding segmentation algorithm,” College of Information Science & Electronic Technology, Tech. Rep., Jan. 2015.
- [11] Y. Zhang, X. Wu, and J. You, “A benchmark dataset and baseline methods for rock microstructure interpretation in sem images,” *Scientific Data*, vol. 12, no. 1, p. 1671, 2025.
- [12] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for

- biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015, pp. 234–241.
- [13] Q. Yasin, B. Liu, M. Sun, G. M. Sohail, A. Ismail, M. Majdanski, N. Golsanami, Y. Ma, and X. Fu, “Automatic pore structure analysis in organic-rich shale using fib-sem and attention u-net,” *Fuel*, vol. 358, p. 130161, 2024.
 - [14] Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3d u-net: Learning dense volumetric segmentation from sparse annotation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2016, pp. 424–432.
 - [15] J. Kousaka, A. H. Iwane, and Y. Togashi, “Automated cell structure extraction for 3d electron microscopy by deep learning,” *Scientific Reports*, vol. 15, no. 1, p. 17481, 2025.
 - [16] Ultralytics, “Yolov8,” Oct. 2025. [Online]. Available: <https://docs.ultralytics.com/models/yolov8/>
 - [17] K. Ferji, “Detectnano: Deep learning detection in tem images for high-throughput nanostructure characterization,” *Nanoscale*, vol. 17, pp. 18 777–18 786, 2025.
 - [18] K. Ruzaeva, K. Govind, M. Legros, and S. Sandfeld, “Instance segmentation of dislocations in tem images,” *arXiv*, 2023.
 - [19] X. Yu, B. Hu, W. Jiang, J. Wan, X. Yang, N. Liu, and X. Dong, “Enhanced yolov8 for industrial polymer films: A semi-supervised framework for micron-scale defect detection,” *Frontiers in Artificial Intelligence*, vol. 8, 2025.
 - [20] B. Vemuri, S. Sahni, F. Chen, C. Kapoor, C. Leonard, and J. Fkzsimmers, “Lossless image compression,” Mar. 2002.
 - [21] N. Annalakshmi, “Lossy image compression techniques,” *International Journal of Computer Applications*, vol. 183, pp. 30–34, 2021.
 - [22] D. Barman, A. Hasnat, and R. Nag, “An introduction to autoencoders,” Feb. 2022.
 - [23] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Learned lossless image compression with a hyperprior and discretized gaussian mixture likelihoods,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 2158–2162.
 - [24] S. Chen, W. Wang, and S. J. Pan, “Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/f8e59f4b2fe7c5705bf878bbd494ccdf-Paper.pdf
 - [25] M. Schäfer, S. Pientka, J. Pfaff, H. Schwarz, D. Marpe, and T. Wiegand, “Rate-

- distortion-optimization for deep image compression,” in *IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3737–3741.
- [26] J. Ascenso and E. Upenik, “White paper on jpeg ai scope and framework,” JPEG.org, Tech. Rep., 2021.
- [27] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Deep convolutional autoencoder-based lossy image compression,” in *Picture Coding Symposium (PCS)*, 2018, pp. 253–257.
- [28] J. Lee, S. Cho, and S.-K. Beack, “Context-adaptive entropy model for end-to-end optimized image compression,” 2018.
- [29] JPEG AI (ISO/IEC 6048), “Final call for proposals for jpeg ai,” in *ISO/IEC JTC 1/SC 29/WG 1*, 2022.
- [30] S. Esenlik, Y. Wu, Z. Zhang, Y.-K. Wang, K. Zhang, L. Zhang, J. Ascenso, and S. Liu, “An overview of the jpeg ai learning-based image coding standard,” 2025.
- [31] N. Nešić, X. Heiligenstein, L. Zopf, V. Blüml, K. Keuenhof, M. Wagner, J. Höög, H. Qi, Z. Li, G. Tsaramirsis, C. Peddie, M. Stojmenovic, and A. Walter, “Automated segmentation of cell organelles in volume electron microscopy using deep learning,” *Microscopy Research and Technique*, vol. 87, pp. 1718–1732, 2024.
- [32] B. Forman and A. MacDonald, “Hela cells: Key discoveries and the science of their immortality,” Technology Networks, Tech. Rep., Sep. 2024.
- [33] T. Faber, J. T. McConville, and A. Lamprecht, “Focused ion beam-scanning electron microscopy provides novel insights of drug delivery phenomena,” *Journal of Controlled Release*, vol. 366, pp. 312–327, 2024.
- [34] M. Luckner and G. Wanner, “From light microscopy to analytical scanning electron microscopy (sem) and focused ion beam (fib)/sem in biology: Fixed coordinates, flat embedding, absolute references,” National Library of Medicine, Tech. Rep., Sep. 2018.
- [35] W. Rasband, “Imagej.” [Online]. Available: <https://imagej.net/ij/docs/intro.html>
- [36] Z. Lu, D. K. Kasaragod, and S. J. Matcher, “Performance comparison between 8- and 14-bit-depth imaging in polarization-sensitive swept-source optical coherence tomography,” National Library of Medicine, Tech. Rep., Mar. 2011.
- [37] W. Wang and Y. Lu, “Analysis of the mean absolute error (mae) and the root mean square error (rmse) in assessing rounding model,” *IOP Conference Series: Materials Science and Engineering*, vol. 324, p. 012049, 2018.
- [38] X. Wang, B. Cai, Z. Jia, Y. Chen, S. Guo, Z. Liu, X. Wan, F. Zhang, and B. Hu, “Mitostructseg: A comprehensive platform for mitochondrial

- structure segmentation and analysis,” *bioRxiv*, 2024. [Online]. Available: <https://www.biorxiv.org/content/early/2024/07/02/2024.06.28.601295>
- [39] Q. Ba, G. Raghavan, K. Kiselyov, and G. Yang, “Whole-cell scale dynamic organization of lysosomes revealed by spatial statistical analysis,” National Library of Medicine, Tech. Rep., Jun. 2018.
- [40] M. Jovic, M. Sharma, J. Rahajeng, and S. Caplan, “The early endosome: A busy sorting station for proteins at the crossroads,” National Library of Medicine, Tech. Rep., Jan. 2010.
- [41] C. C. Scott, F. Vacca, and J. Gruenberg, “Endosome maturation, transport and functions,” National Library of Medicine, Tech. Rep., Jul. 2014.
- [42] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft coco: Common objects in context,” 2014. [Online]. Available: <https://arxiv.org/abs/1405.0312>
- [43] G. Bradski, “The opencv library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [44] T. DALSA, “Sherlock: Machine vision software,” 2024. [Online]. Available: <https://www.teledynevisionsolutions.com/products/sherlock>
- [45] Ultralytics, “Yolo26,” Mar. 2026. [Online]. Available: <https://docs.ultralytics.com/pt/models/yolo26/>
- [46] S. Mukherjee, “Object detection with yolov8 advanced capabilities,” Apr. 2025. [Online]. Available: <https://www.digitalocean.com/community/tutorials/yolov8-a-revolutionary-advancement-in-object-detection-2>
- [47] V. González-Ruiz, M. R. Fernández-Fernández, and J. J. Fernández, “Structure-preserving gaussian denoising of fib-sem volumes,” 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0304399122001930>
- [48] D. Seol, J. Choi, C. Kim, and S. Hong, “Alleviating class-imbalance data of semiconductor equipment anomaly detection study,” *Electronics*, vol. 12, p. 585, 2023.
- [49] P. Virtanen, R. Gommers, T. E. Oliphant *et al.*, “SciPy 1.0: Fundamental algorithms for scientific computing in python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [50] N. U. Huda, “Precision recall curves,” Sep. 2025. [Online]. Available: <https://www.blog.trainindata.com/precision-recall-curves/>
- [51] Z. Jian, T. Song, Z. Zhang, Z. Ai, H. Zhao, M. Tang, and K. Liu, “Deep learning method for detecting fluorescence spots in cancer diagnostics via fluorescence in situ hybridization,” *Scientific Reports*, vol. 14, no. 1, p. 27231, 2024.
- [52] Ultralytics, “Model training with ultralytics yolo,” Jan. 2026. [Online]. Available: <https://docs.ultralytics.com/modes/train/#train-settings>
- [53] —, “Data augmentation using ultralytics yolo,” Jan. 2026. [On-

- line]. Available: <https://docs.ultralytics.com/guides/yolo-data-augmentation/#classification-specific-augmentations>
- [54] R. Starosolski, "Hybrid adaptive lossless image compression based on discrete wavelet transform," *Entropy*, vol. 22, no. 7, p. 751, 2020.
- [55] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," 2003. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1292216>
- [56] M. Nadipally, "Optimization of methods for image-texture segmentation using ant colony optimization," 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/B9780128155530000021>
- [57] H. R. Sheikh and A. C. Bovik, "Image information and visual quality." [Online]. Available: <http://live.ece.utexas.edu/research/quality/VIF.htm>
- [58] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: A feature similarity index for image quality assessment." [Online]. Available: https://www4.comp.polyu.edu.hk/~cslzhang/IQA/TIP_IQA_FSIM.pdf
- [59] A. Hepburn, V. Laparra, R. McConville, and R. Santos-Rodríguez, "Enforcing perceptual consistency on generative adversarial networks by using the normalized laplacian pyramid distance," in *NL DL*, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:199551861>
- [60] Z. Wang and Q. Li, "Information content weighting for perceptual image quality assessment." [Online]. Available: <https://ece.uwaterloo.ca/~z70wang/publications/IWSSIM.pdf>
- [61] Netflix, "Vmaf: Video multi-method assessment fusion." [Online]. Available: <https://github.com/Netflix/vmaf>
- [62] K. Egiazarian, J. Astola, V. Lukin, F. Battisti, and M. Carli, "A new full-reference quality metric based on hvs," 2006. [Online]. Available: https://www.researchgate.net/publication/251229783_A_NEW_FULL-REFERENCE_QUALITY_METRICS_BASED_ON_HVS
- [63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.
- [64] L. Wang, "A survey on iqa," 2021.

ANNEXES

Annex B - Compression Results for Base operation point and tools off - GPU-LH mode

Table 7.1: Average values of metrics for Base operation point on tools off - GPU-LH compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,51	54,44	56,19	0,50	1,00	0,13	0,99	88,07	35,49
50	0,99	0,99	30,85	57,09	56,77	0,48	1,00	0,16	0,99	85,71	33,17
25	0,97	0,97	28,92	57,16	57,98	0,40	0,99	0,21	0,97	75,12	29,34
12	0,96	0,96	27,92	56,01	56,88	0,35	0,99	0,25	0,95	66,45	27,03
6	0,93	0,92	26,55	56,65	57,35	0,28	0,97	0,32	0,91	46,43	23,83

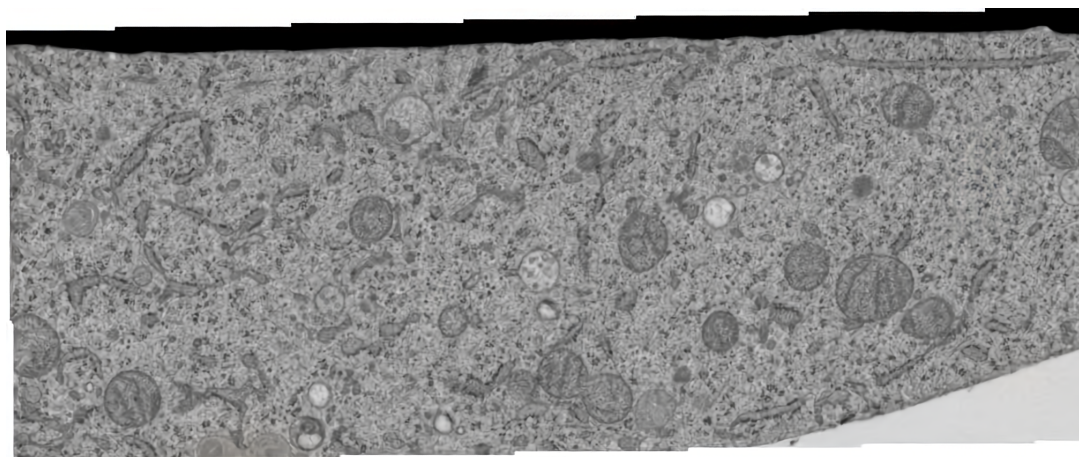


Figure 7.2: Reconstructed image of Base operation point on tools off – GPU LH compression at 6% bit rate

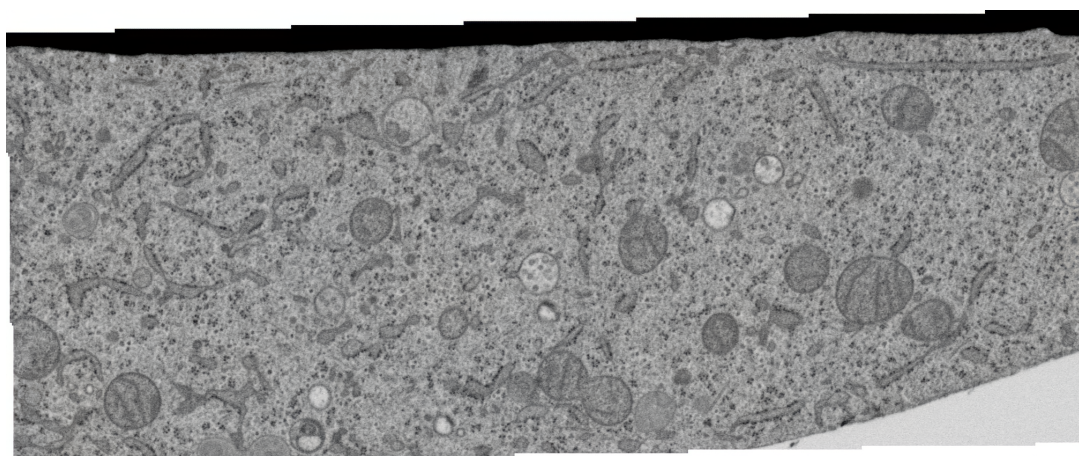


Figure 7.3: Reconstructed image of Base operation point on tools off – GPU LH compression at 75% bit rate

Annex C - Compression Results for Base operation point and only ResVarScale mode

Table 7.2: Average values of metrics for Base operation point on only ResVarScale compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,98	0,98	29,39	57,14	56,78	0,43	1,00	0,17	0,98	82,38	31,91
50	0,99	0,99	30,93	57,09	56,77	0,48	1,00	0,15	0,99	87,11	33,79
25	0,97	0,97	28,96	57,09	58,02	0,40	0,99	0,20	0,97	77,42	29,66
12	0,96	0,96	27,96	56,00	56,86	0,36	0,99	0,24	0,96	69,28	27,20
6	0,91	0,91	26,10	56,86	57,74	0,26	0,96	0,35	0,89	42,98	22,88

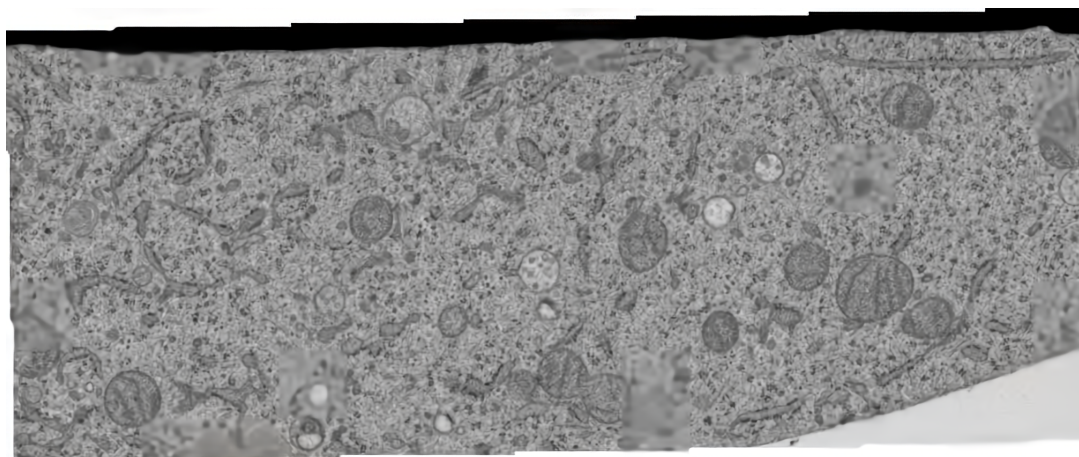


Figure 7.4: Reconstructed image of Base operation point on only ResVarScale compression at 6% bit rate

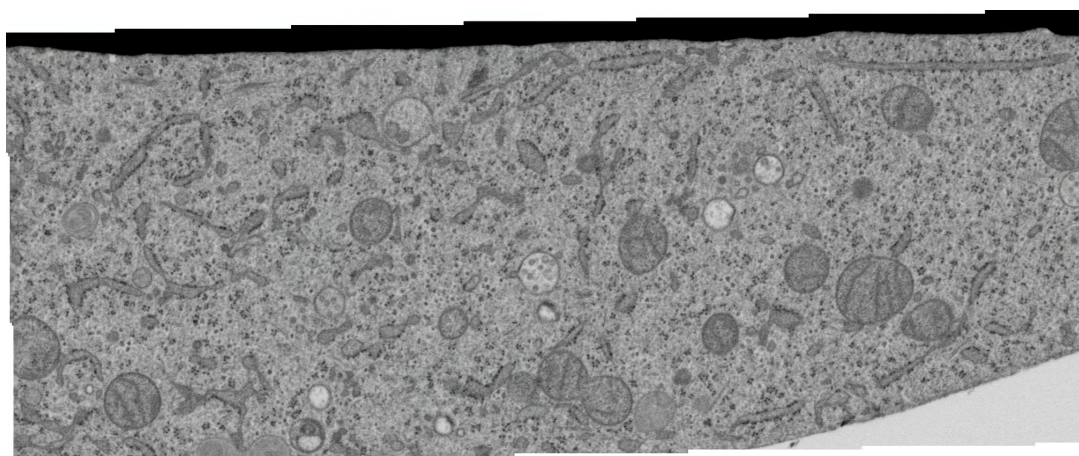


Figure 7.5: Reconstructed image of Base operation point on only ResVarScale compression at 75% bit rate

Annex D - Compression Results for Base operation point and only ICCI mode

Table 7.3: Average values of metrics for Base operation point on only ICCI compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,55	54,26	57,34	0,51	1,00	0,13	0,99	89,66	35,56
50	0,99	0,99	30,85	57,13	57,86	0,48	1,00	0,16	0,99	85,75	33,17
25	0,97	0,97	28,91	57,17	57,99	0,40	0,99	0,21	0,97	75,42	29,34
12	0,96	0,96	27,92	56,60	57,03	0,35	0,99	0,25	0,95	66,63	27,02
6	0,93	0,92	26,52	57,64	56,95	0,28	0,97	0,33	0,91	47,60	23,75

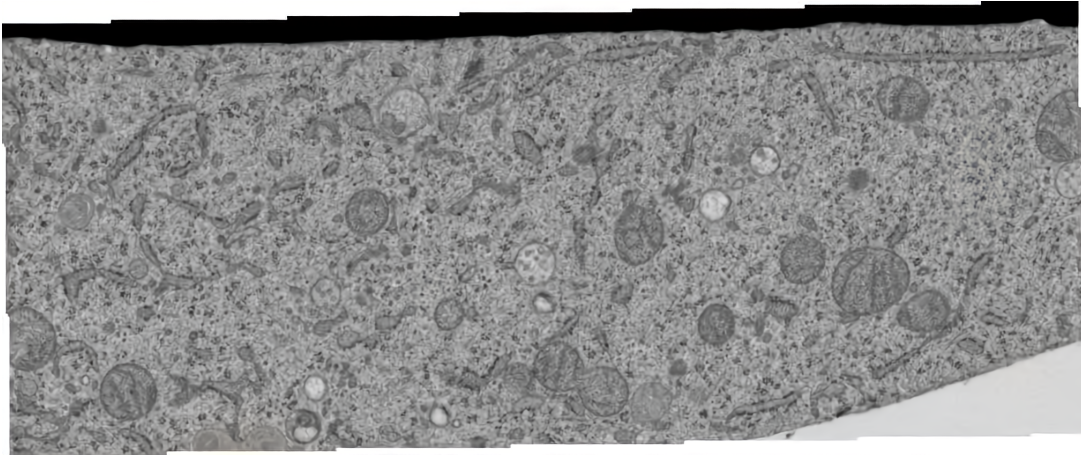


Figure 7.6: Reconstructed image of Base operation point on only ICCI compression at 6% bit rate

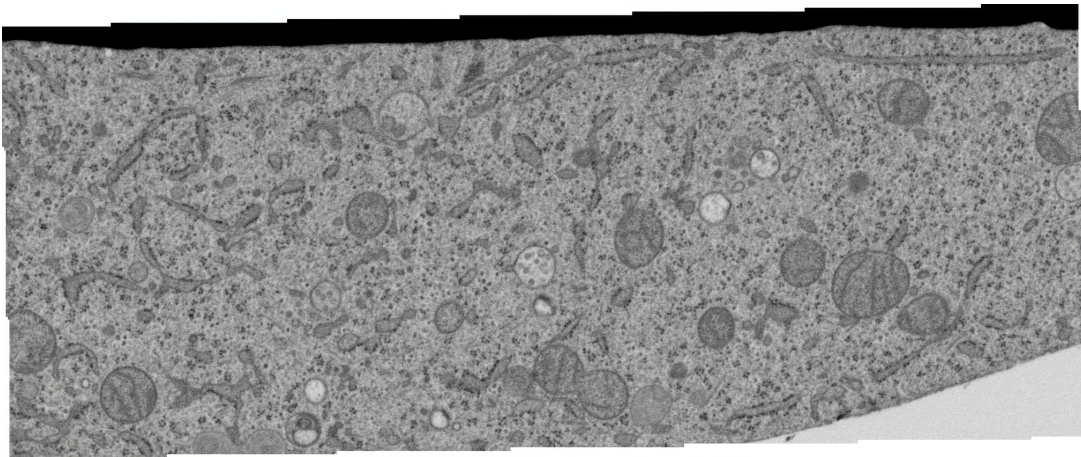


Figure 7.7: Reconstructed image of Base operation point on only ICCI compression at 75% bit rate

Annex E - Compression Results for Base operation point and only LSBS mode

Table 7.4: Average values of metrics for Base operation point on only LSBS compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,98	0,98	29,38	57,13	56,81	0,43	1,00	0,17	0,98	83,78	31,84
50	0,99	0,99	30,82	57,09	56,80	0,48	1,00	0,16	0,98	86,99	33,06
25	0,97	0,97	28,88	57,08	58,07	0,40	0,99	0,21	0,97	76,25	29,21
12	0,96	0,96	27,87	55,93	56,88	0,35	0,99	0,25	0,95	67,13	26,88
6	0,91	0,90	26,03	56,75	57,44	0,26	0,96	0,36	0,89	41,13	22,77

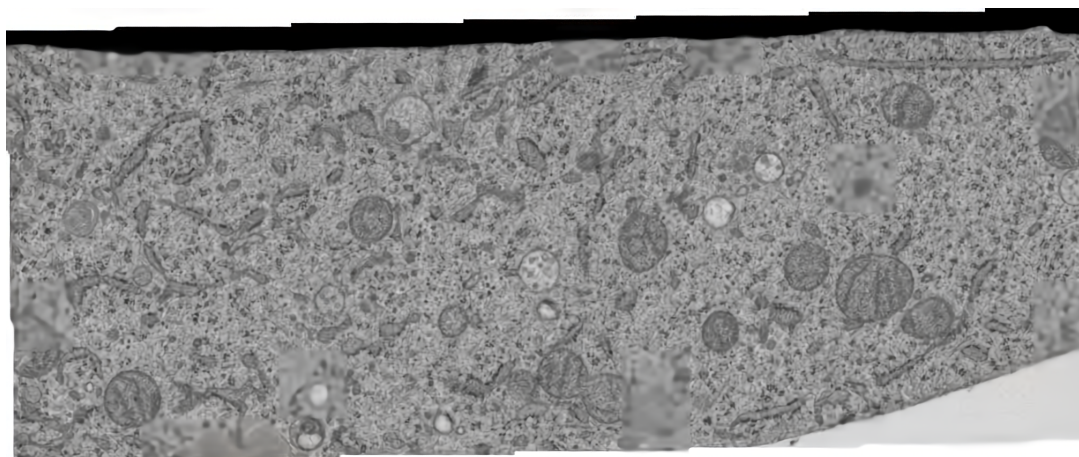


Figure 7.8: Reconstructed image of Base operation point on only LSBS compression at 6% bit rate

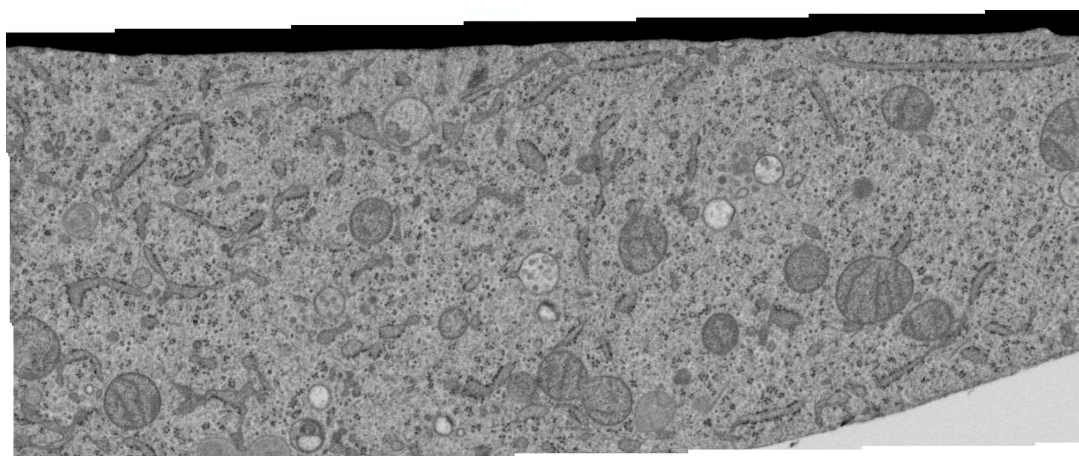


Figure 7.9: Reconstructed image of Base operation point on only LSBS compression at 75% bit rate

Annex F - Compression Results for Base operation point and only ECThread8 mode

Table 7.5: Average values of metrics for Base operation point on only ECThread8 compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,51	54,44	56,19	0,50	1,00	0,13	0,99	88,07	35,49
50	0,99	0,99	30,85	57,09	56,77	0,48	1,00	0,16	0,99	85,71	33,17
25	0,97	0,97	28,92	57,16	57,98	0,40	0,99	0,21	0,97	75,12	29,34
12	0,96	0,96	27,92	56,01	56,88	0,35	0,99	0,25	0,95	66,45	27,03
6	0,93	0,92	26,55	56,65	57,35	0,28	0,97	0,32	0,91	46,43	23,83

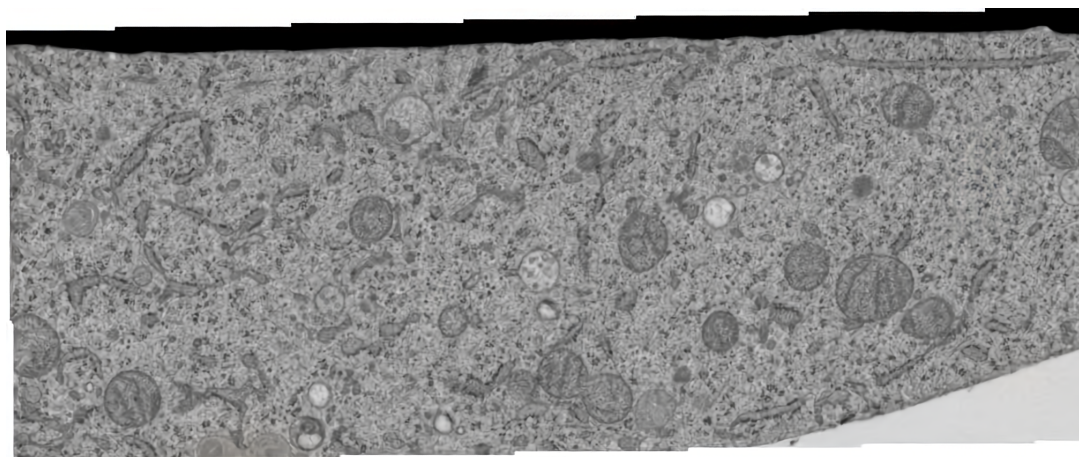


Figure 7.10: Reconstructed image of Base operation point on only ECThread8 compression at 6% bit rate

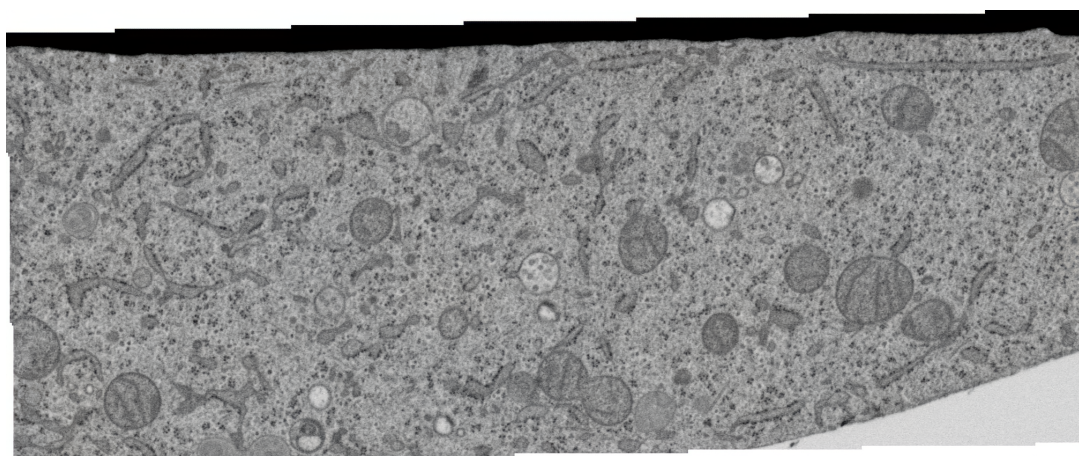


Figure 7.11: Reconstructed image of Base operation point on only ECThread8 compression at 75% bit rate

Annex G - Compression Results for High operation point and tools of - GPU mode

Table 7.6: Average values of metrics for High operation point on tools off - GPU compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,45	61,28	59,59	0,50	1,00	0,13	0,99	88,83	35,77
50	0,99	0,99	30,80	61,28	59,56	0,48	1,00	0,15	0,99	85,99	33,38
25	0,98	0,97	28,92	60,74	61,55	0,40	0,99	0,21	0,97	76,76	29,46
12	0,96	0,96	27,93	60,84	60,50	0,36	0,99	0,24	0,96	68,61	27,08
6	0,93	0,92	26,47	49,92	60,18	0,28	0,97	0,32	0,91	49,44	23,71

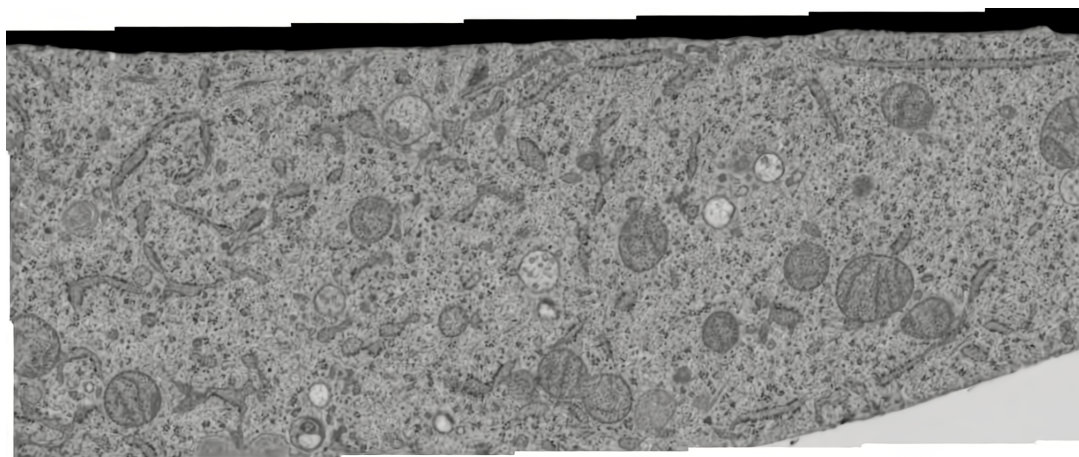


Figure 7.12: Reconstructed image of High operation point on tools off - GPU compression at 6% bit rate

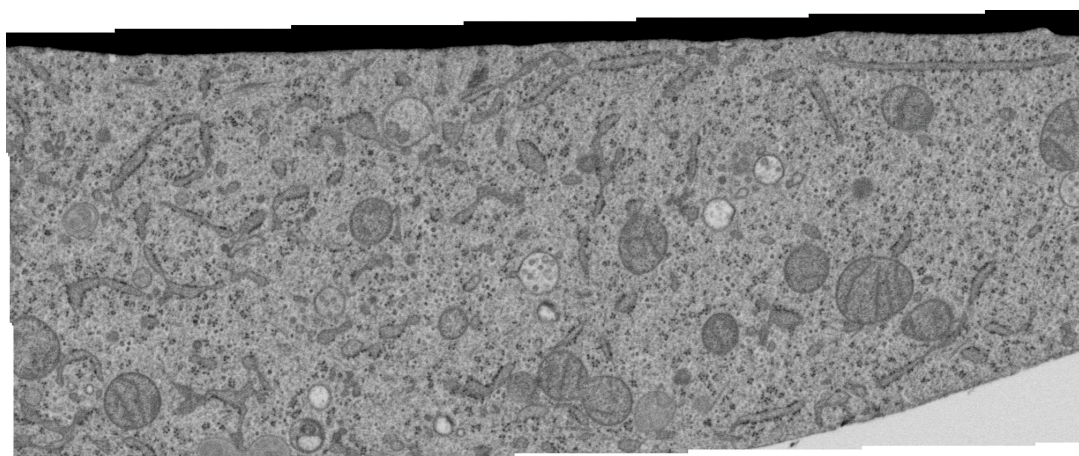


Figure 7.13: Reconstructed image of Base operation point on tools off - GPU compression at 75% bit rate

Annex H - Compression Results for High operation point and tools of - GPU-LH mode

Table 7.7: Average values of metrics for High operation point on tools off - GPU-LH compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,45	61,28	59,59	0,50	1,00	0,13	0,99	88,83	35,77
50	0,99	0,99	30,80	61,28	59,56	0,48	1,00	0,15	0,99	85,99	33,38
25	0,98	0,97	28,92	60,74	61,55	0,40	0,99	0,21	0,97	76,76	29,46
12	0,96	0,96	27,93	60,84	60,50	0,36	0,99	0,24	0,96	68,61	27,08
6	0,93	0,92	26,47	49,92	60,18	0,28	0,97	0,32	0,91	49,44	23,71

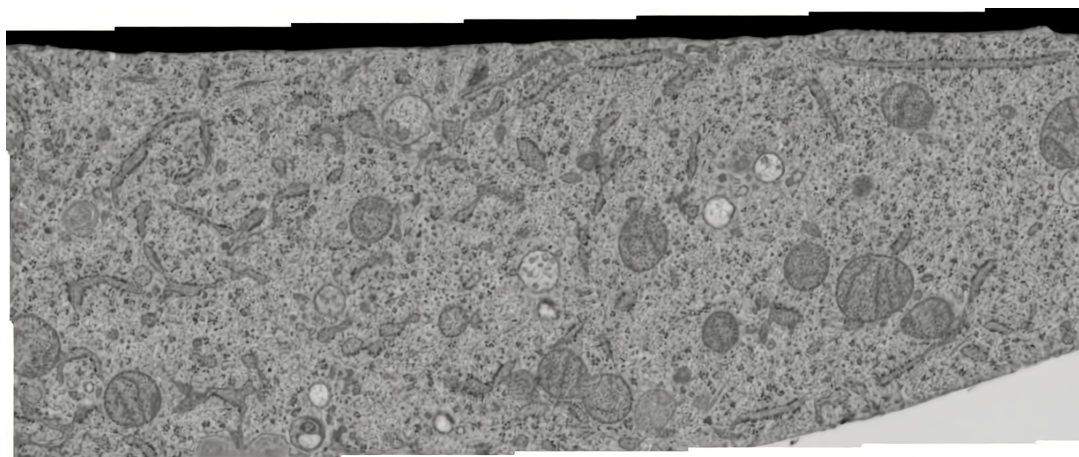


Figure 7.14: Reconstructed image of High operation point on tools off – GPU LH compression at 6% bit rate

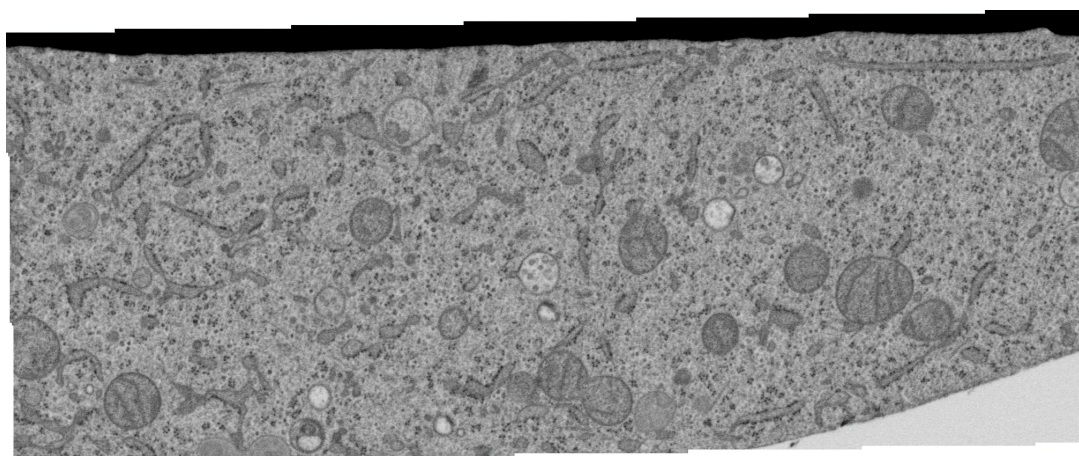


Figure 7.15: Reconstructed image of High operation point on tools off – GPU LH compression at 75% bit rate

Annex I - Compression Results for High operation point and only ResVarScale mode

Table 7.8: Average values of metrics for High operation point on only ResVarScale compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,53	61,28	59,59	0,50	1,00	0,13	0,99	89,38	36,21
50	0,99	0,99	30,97	61,28	59,56	0,48	1,00	0,15	0,99	87,49	34,19
25	0,98	0,98	28,97	62,34	61,43	0,41	0,99	0,20	0,97	78,88	29,81
12	0,96	0,96	28,01	60,86	60,54	0,36	0,99	0,23	0,96	71,30	27,35
6	0,93	0,93	26,54	49,93	60,26	0,28	0,98	0,31	0,91	52,25	23,85

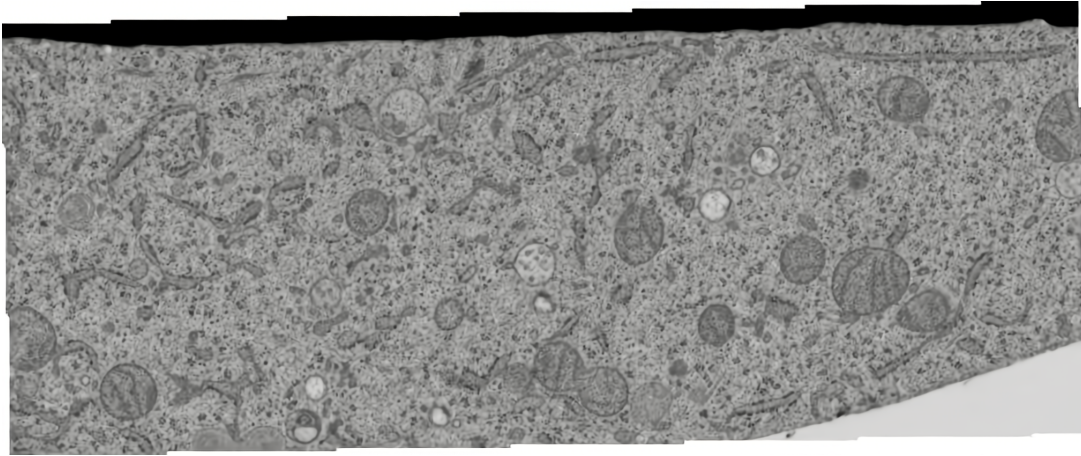


Figure 7.16: Reconstructed image of High operation point on only ResVarScale compression at 6% bit rate

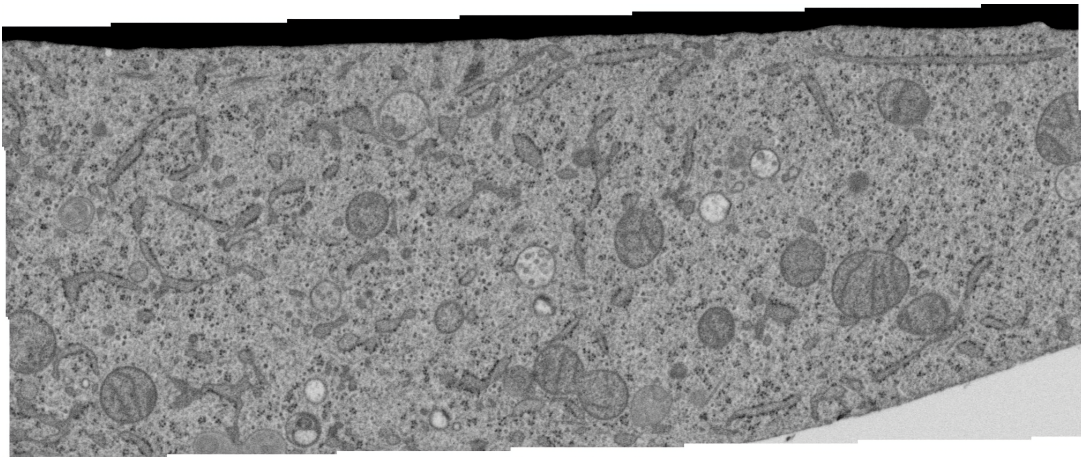


Figure 7.17: Reconstructed image of High operation point on only ResVarScale compression at 75% bit rate

Annex J - Compression Results for High operation point and only ICCI mode

Table 7.9: Average values of metrics for High operation point on only ICCI compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,44	61,32	61,33	0,50	1,00	0,13	0,99	90,22	35,69
50	0,99	0,99	30,80	61,36	61,31	0,48	1,00	0,15	0,99	85,99	33,38
25	0,98	0,97	28,92	61,61	61,56	0,40	0,99	0,21	0,97	76,77	29,46
12	0,96	0,96	27,95	60,37	59,59	0,36	0,99	0,24	0,96	68,42	27,10
6	0,93	0,92	26,44	56,45	59,51	0,28	0,97	0,32	0,91	50,26	23,66

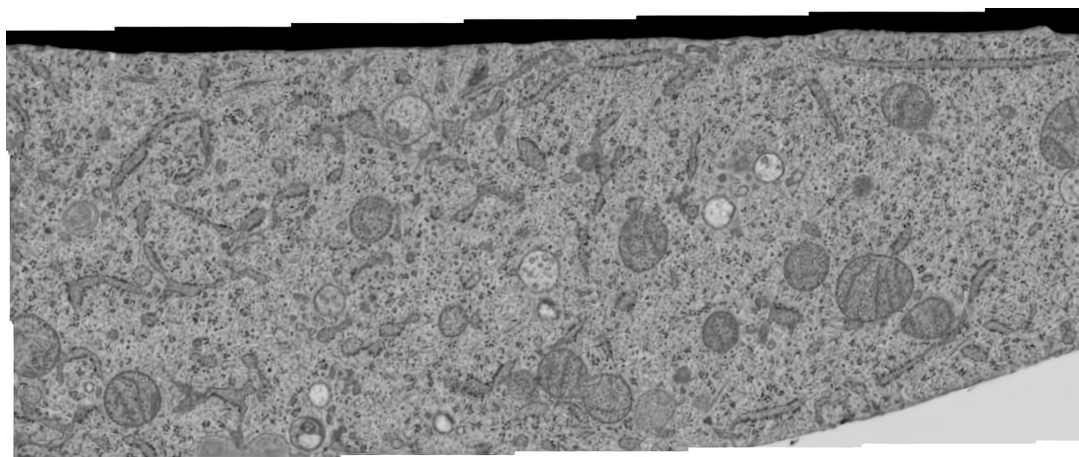


Figure 7.18: Reconstructed image of High operation point on only ICCI compression at 6% bit rate

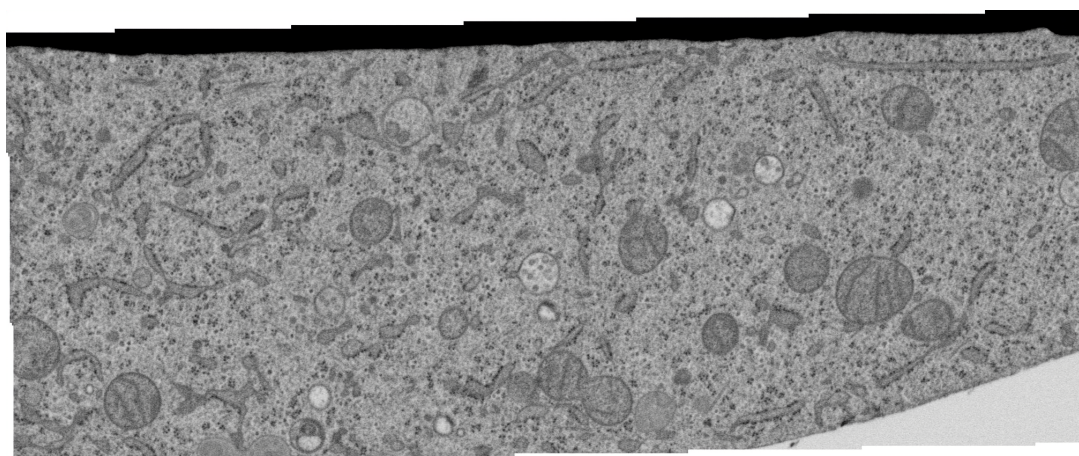


Figure 7.19: Reconstructed image of High operation point on only ICCI compression at 75% bit rate

Annex K - Compression Results for High operation point and only LSBS mode

Table 7.10: Average values of metrics for High operation point on only LSBS compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,45	61,32	59,68	0,50	1,00	0,13	0,99	90,64	35,73
50	0,99	0,99	30,79	61,33	59,65	0,48	1,00	0,16	0,99	87,44	33,31
25	0,98	0,97	28,92	62,35	61,36	0,40	0,99	0,21	0,97	78,22	29,43
12	0,96	0,96	27,93	60,77	60,54	0,36	0,99	0,24	0,96	69,84	27,06
6	0,93	0,92	26,46	49,69	60,20	0,28	0,97	0,32	0,91	50,25	23,70

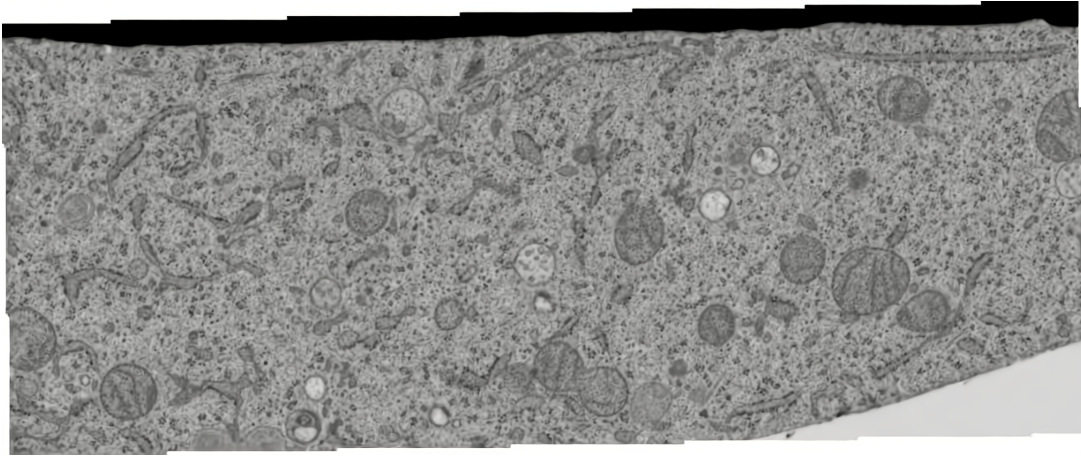


Figure 7.20: Reconstructed image of High operation point on only LSBs compression at 6% bit rate

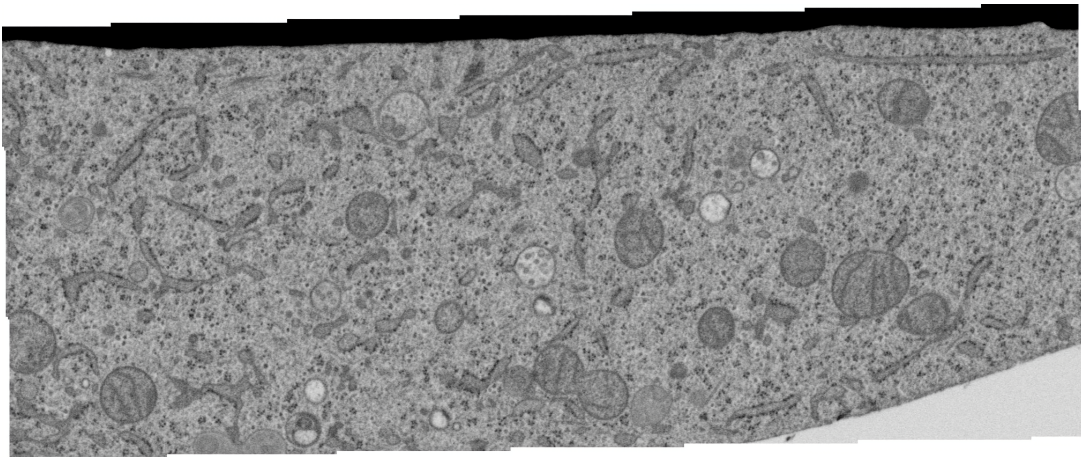


Figure 7.21: Reconstructed image of High operation point on only LSBs compression at 75% bit rate

Annex L - Compression Results for High operation point and only ECThread8 mode

Table 7.11: Average values of metrics for High operation point on only ECThread8 compression

Target Bit Rate	MS-SSIM Torch	MS-SSIM IQA	PSNR Y	PSNR U	PSNR V	VIF	FSIM	NLPD	IW-SSIM	VMAF	PSNR HVS
75	0,99	0,99	31,45	61,28	59,59	0,50	1,00	0,13	0,99	88,83	35,77
50	0,99	0,99	30,80	61,28	59,56	0,48	1,00	0,15	0,99	85,99	33,38
25	0,98	0,97	28,92	60,74	61,55	0,40	0,99	0,21	0,97	76,76	29,46
12	0,96	0,96	27,93	60,84	60,50	0,36	0,99	0,24	0,96	68,61	27,08
6	0,93	0,92	26,47	49,92	60,18	0,28	0,97	0,32	0,91	49,44	23,71

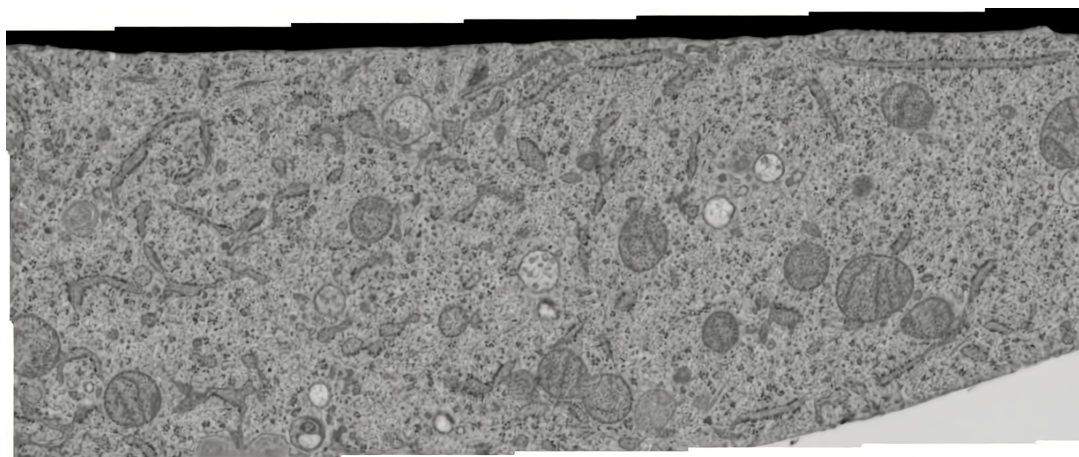


Figure 7.22: Reconstructed image of High operation point on only ECThread8 compression at 6% bit rate

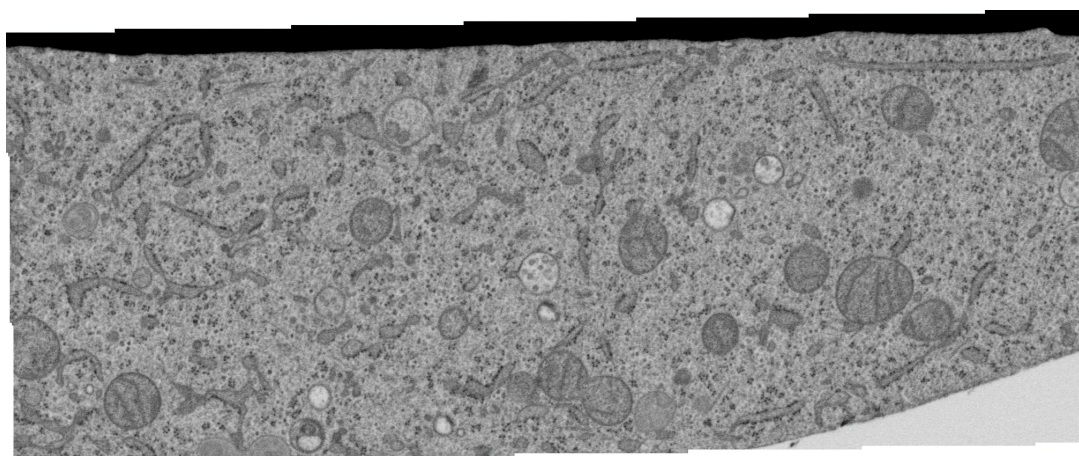


Figure 7.23: Reconstructed image of High operation point on only ECThread8 compression at 75% bit rate



**Instituto Superior
de Engenharia**

Politécnico de Coimbra