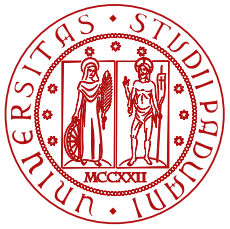


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Semantic Segmentation of Cavities in Electron Microscopy Images using Mask R-CNNs

Master Candidate

Andrea Maci

Student ID 2063146

Supervisor

Prof. Loris Nanni

University of Padova

Academic Year
2023/2024

Abstract

The analysis of the conditions of materials is a key element for granting safety and stability in several environments. In particular, localizing cavities in materials, for instance metals exposed to extreme conditions, helps to prevent greater damages in potentially unsafe environments. In this thesis, various analysis methods are presented by scanning images of metals using neural networks in the form of Mask R-CNNs, that is, Mask Region-based Convolutional Neural Networks. This neural network architecture is exploited to perform a semantic segmentation of the electronic microscopy images; the segmentation unveils the cavities of irradiated metal alloys instead of manually checking the microscopy images. In this study, the goal is to reproduce and to develop new techniques to attempt to improve the results obtained in the paper "Materials swelling revealed through automated semantic segmentation of cavities in electron microscopy images" [1], by implementing a custom loss function that takes into account the estimated material swelling, or by adopting a standard segmentation model, such as DeepLabV3+, for separating the background from the cavities. This study also showcases the usefulness and ease of access of MATLAB for all of the aforementioned deep learning applications and implementations.

Sommario

L'analisi delle condizioni di materiali è un elemento chiave per garantire sicurezza e stabilità in diversi ambienti. In particolar modo, localizzare cavità nelle superfici di materiali, ad esempio metalli esposti a condizioni estremi, aiuta a prevenire danni maggiori in ambienti potenzialmente poco sicuri. In questa tesi, svariati metodi di analisi vengono presentati, attraverso una scansione di immagini di metalli tramite reti neurali nella forma di Mask R-CNN, ovvero una Mask Region-based Convolutional Neural Network. Questa rete neurale è in grado di eseguire la segmentazione semantica di immagini al microscopio elettronico; in questo caso, la segmentazione rileva eventuali cavità in leghe metalliche irradiate in modo automatico, invece di eseguire controlli manuali. In questo studio, lo scopo è sviluppare nuoev tecniche nel tentativo di migliore i risultati ottenuti nell'articolo scientifico "Materials swelling revealed through automated semantic segmentation of cavities in electron microscopy images" [1], implementando una loss function personalizzata che prende in considerazione il rigonfiamento dei metalli, oppure adottando altre metodologie standard di segmentazione come DeepLabV3+, per separare lo sfondo dalle cavità. Questo studio, infine, dimostra l'utilità e la facilità di accesso a MATLAB per tutte le tecniche e implementazioni di deep learning menzionate precedentemente.

Contents

List of Figures

List of Tables

List of Acronyms

CNN Convolutional Neural Network

R-CNN Regional Convolutional Neural Network

SOTA state-of-the-art

DSCNN Dynamic Segmentation Convolutional Neural Network

S/TEM Scanning/Transmission Electron Microscopy

COCO Common Objects in Context

ResNet Residual Network

RoI Region of Interest

JSON JavaScript Object Notation

PNG Portable Network Graphics

GPU Graphics Processing Unit

CNL Canadian Nuclear Laboratory

NOME Nuclear Oriented Materials & Examination

RPN Region Proposal Network

SGDM Stochastic Gradient Descent with Momentum

FPN Feature Pyramid Network

IoU Intersection over Union

MAE Mean Absolute Error

1

Introduction

Nuclear materials science is essential for the construction and operation of efficient and reliable nuclear reactors. This field focuses on the study of alloy swelling, a phenomenon that occurs when metal alloys are subjected to radiation. Such swelling can compromise the structural integrity of reactor components, potentially leading to catastrophic failures. Traditionally, the measurement of radiation-induced swelling in alloys was performed manually by experts who examined electron microscopy images to identify and quantify defects such as cavities and voids. However, this manual method is time-consuming, subjective, and lacks scalability, especially with the increasing volume of high-resolution microscopy data generated by modern devices. The aim of this paper is to study new techniques to automatize this process and to potentially improve beyond human accuracy.

Recent advancements in computer vision, particularly deep learning techniques, provide promising solutions to these manual challenges. Deep learning models, such as Convolutional Neural Networks (CNNs), have revolutionized image analysis by enabling automated and highly accurate object detection and segmentation. Notably, the Mask R-CNN has proven to be a powerful tool for instance segmentation, providing pixel-level annotations of objects while being able to distinguish object-per-object within the provided images. Leveraging these advanced models, it is possible to automate the detection and quantification of nanoscale cavities in irradiated alloys, thereby enhancing the efficiency and accuracy of swelling assessments.

This thesis builds on the initial research presented in our foundational paper,

"Materials swelling revealed through automated semantic segmentation of cavities in electron microscopy images" [1]. The study demonstrated the potential of using Mask R-CNN to detect and measure cavities in irradiated alloys, achieving encouraging results in terms of precision, recall, and F1 score. Nonetheless, there remains significant room for improvement, particularly in incorporating additional segmentation models and developing custom loss functions tailored to specific material characteristics, such as swelling in this case study.

This thesis aims to advance nuclear material studies by creating more precise and effective tools for assessing radiation-induced swelling. These innovations could significantly improve nuclear reactor material design and operational safety, ensuring more reliable performance under irradiation. Additionally, the study's methodology and findings can be applied to other fields requiring precise image-based defect assessment, highlighting the broad applicability of deep learning in scientific research. The use of MATLAB as a primary platform underscores the research's practicality and accessibility, making it highly relevant to the engineering community.

MATLAB is a powerful and versatile tool widely used in engineering and scientific research, it provides a vast library of built-in functions and toolboxes that cater to various mathematical, statistical, and engineering computations. These tools facilitate complex data analysis, numerical modeling, and plenty of deep learning implementations. To advance the state-of-the-art (SOTA) technologies in automated swelling identification, the following work, exploiting MATLAB, proposes to:

- Implement a baseline Mask R-CNN model to begin instance segmentation.
- Tune several parameters to seek for improvements. These parameters range from network specific properties to neural network training options
- Explore the potential of DeepLabV3+ for background segmentation to attempt to improve the Mask R-CNN training routine.
- Implement a custom loss function for training that not only takes into account the pixel classification error and mask size error, but also the ad-hoc estimated swelling of the materials.

Moreover, most of the MATLAB code was run in the DEI Computing Cluster "Blade", kindly provided by the university. A blade computing cluster is highly beneficial for executing MATLAB code, particularly when in need of powerful GPUs. Such clusters consist of multiple blade servers, each equipped with

high-performance GPUs, which significantly enhance computational capabilities. This setup is ideal for running large-scale simulations, more specifically for the training of deep neural networks. The parallel processing power of GPUs accelerates tasks like deep learning, image processing, and numerical modeling, drastically reducing execution time. Additionally, MATLAB's Parallel Computing Toolbox and GPU support allow for anyone to seamlessly distribute workloads across the cluster, optimizing resource utilization and achieving faster results. This makes the Blade computing cluster an invaluable asset for any computational challenge encountered during this thesis.

2

Research Background

2.1 Alloy Swelling in Nuclear Reactor Environments

Alloy swelling in nuclear reactors is a critical issue affecting the durability and safety of reactor components. Irradiation of metal alloys in reactor cores and surrounding areas induces defects such as dislocation loops, precipitates, and cavities. These cavities, termed voids when gas-free and bubbles when gas-filled, deteriorate mechanical properties by increasing hardness, brittleness, and causing swelling. Hence, understanding when a material needs to be replaced due to these damages is needless to say, crucial.

Neutron irradiation causes alloy swelling through a complex process involving various mechanisms. The enlargement of voids, exacerbated by the presence of helium, leads to the formation of bubbles within the material's microstructure. This can result in swelling and grain boundary embrittlement, compromising the integrity of reactor materials. While neutron irradiation increases the strength of these materials, it also significantly reduces their ductility, causing radiation hardening and uneven plastic deformation. According to Ghoniem et al. [2], exposure of structural materials to neutron irradiation creates various atomic-scale defects. These defects often enhance the material's strength but greatly reduce its ability to stretch or deform. Understanding these interactions and their outcomes is essential for maintaining the structural stability of reactors and ensuring the safe and efficient operation of nuclear power

2.2 Measuring Swelling Using Electron Microscopy

Measuring the extent of swelling in nuclear reactor materials using electron microscopy, especially Scanning/Transmission Electron Microscopy (S/TEM), is vital for understanding material degradation due to irradiation. TEM imaging conditions and sample properties can cause variations that lead to perceived swelling rather than precise measurements. Accurate quantification of swelling is essential to ensure the structural stability and safety of nuclear reactors. Quantification challenges arise due to phase shifts caused by differences in the mean inner potential between cavities and the surrounding crystal. These shifts significantly impact the observed size of cavities under various imaging conditions. The foundational paper emphasizes that multi-slice simulations have demonstrated considerable changes in the perceived size of cavities when using Fresnel contrast imaging at different underfocus levels [1].

To observe the distinct white centroid and dark fringe contrast of holes while minimizing black fringe displacement, it is essential to maintain a small underfocus (less than 1 micrometre). This technique is crucial for the precise measurement of cavities, particularly those smaller than 10 nanometres. However, the process is complicated by errors resulting from imaging parameters such as resolution, human measurement accuracy, sample tilt, and changes in background contrast.

Conventional techniques involve manually counting and measuring individual cavities using software like ImageJ. This approach is hindered by the time-consuming nature of preparing TEM samples and identifying cavities. Advances in sample preparation methods, such as high-throughput focused ion beam (FIB) procedures and flash polishing, have alleviated some limitations. Modern TEM devices have improved data collection rates, producing higher resolution images and larger datasets. However, the manual labeling of cavities remains a significant challenge. There is a clear need for automated techniques capable of quickly processing extensive TEM datasets.

Utilizing machine learning techniques is essential for addressing these challenges, with automation playing a key role. Convolutional Neural Networks (CNNs) have proven to be highly effective for automated microstructure analysis. These networks can accurately and reliably identify specific microstructural features from images in a cost-efficient and consistent manner.

Anderson et al. [3] employed a region-based convolutional neural network

to analyze helium bubbles in irradiated X750 Ni alloys. They successfully pinpointed the locations of these bubbles in underfocused TEM images. This approach highlights the potential for efficient data analytics in materials science. Furthermore, understanding the stability of voids caused by irradiation is crucial for effectively managing material swelling. A study by Chen et al. [4] provides valuable insights into the mechanisms governing the stability transition of voids. Future initiatives involve the improved integration of automated analysis tools to enable the efficient processing and analysis of large datasets. These advancements will enhance our understanding of material behavior under radiation exposure and aid in developing and evaluating new materials and production methods to improve performance and safety in nuclear reactors. Ultimately, quantifying swelling in nuclear reactor materials using electron microscopy is crucial for understanding the deterioration caused by irradiation. Automation and machine learning significantly enhance this process, providing more precise, reliable, and comprehensive assessments of material swelling. These technological advancements are vital for maintaining the structural stability of reactors and ensuring the safe and efficient operation of nuclear power facilities.

2.3 Deep Learning in Material Science

In the past decade, deep learning techniques have revolutionized numerous scientific fields, including material science. Deep learning, a subset of machine learning, employs computational models with multiple layers to learn data representations at varying levels of abstraction. Convolutional Neural Networks (CNNs) have been particularly influential, excelling in image classification, object detection, and semantic segmentation tasks. These advancements have profoundly impacted material science, especially in analyzing and characterizing the microscopic structures of materials. Multiple studies have demonstrated the effectiveness of deep learning in material science. For instance, Stepashkin et al. [5] investigated thermoplastic unidirectional carbon fiber-polysulfone composites, using CNN-based models to predict the tensile strength of these materials. Their algorithms achieved a notably accurate prediction with a Spearman correlation coefficient of 0.988, showcasing the capability of neural networks to precisely predict material properties by analyzing microstructural attributes.

Mantawy et al. [6] conducted a significant study using CNNs to predict

fractures caused by low-cycle fatigue in reinforcing bars. By converting strain time series data into images, the CNN model achieved a testing accuracy of over 96%, demonstrating its effectiveness in forecasting damage.

The field of deep learning continues to advance, introducing new techniques for material analysis. Roberts et al. [7] developed an innovative convolutional neural network architecture called DefectSegNet for the semantic segmentation of crystallographic defects in steels. This model exhibited exceptional pixel-level accuracy for various defect types, surpassing human expert analysis in both speed and consistency. Taller et al. [8] have also demonstrated the application of Dynamic Segmentation Convolutional Neural Networks (DSCNN) to rapidly and quantitatively identify microstructural features in materials.

Furthermore, Lin et al. [9] demonstrated the use of CNNs to study spatial correlations among various regions in cementitious materials. This approach provided new insights into the detailed information and spatial relationships within materials, enhancing understanding of material dynamics and enabling precise evaluations and reconstructions.

Sainju et al. [10] have achieved notable advancements in interpreting in-situ TEM videos with their development of DefectTrack, a one-shot multi-object tracking model. This model excels in identifying and tracking clusters of defects in real-time, outperforming human specialists in terms of both accuracy and speed.

State-of-the-art deep learning models such as ResNet50, ResNet101, and VGG16, extensively trained on datasets like ImageNet and COCO, are pivotal in computer vision tasks. These models serve as the basis for advanced frameworks such as Faster R-CNN and Mask R-CNN, enabling precise object detection and segmentation. In material science, these architectures have been adapted to identify and quantify imperfections such as dislocations, precipitates, and voids in electron microscopy images. Over the last decade, deep learning techniques have significantly advanced in this domain [1].

Deep learning has profound implications for material science. Automated analysis techniques reduce the reliance on manual annotations, significantly speeding up data processing and improving the accuracy of material characterization. This capability enables researchers to efficiently handle large datasets, facilitating more comprehensive investigations into the properties and behaviors of materials. Mishra et al. [11] underscore that deep learning models excel in extracting intricate feature information from images, empowering them to

accomplish sophisticated tasks like image classification, object detection, and image segmentation.

2.4 Mask R-CNN model

Instance segmentation models are pivotal in computer vision as they facilitate the detection and accurate delineation of individual objects within an image. In material science, these models are invaluable for analyzing and characterizing microstructures, defects, and other significant features visible in microscopic images.

Mask R-CNN is an advanced model designed for instance segmentation, building upon Faster R-CNN by incorporating a dedicated component to predict segmentation masks for each Region of Interest (RoI). Developed by He et al. [12] in 2017, Mask R-CNN has set new benchmarks in the domains of object detection and segmentation. The research underscores that Mask R-CNN improves upon Faster R-CNN by integrating an additional capability for predicting segmentation masks, thereby combining the advantages of object detection and semantic segmentation within a unified framework. Let's break down the components of this model:

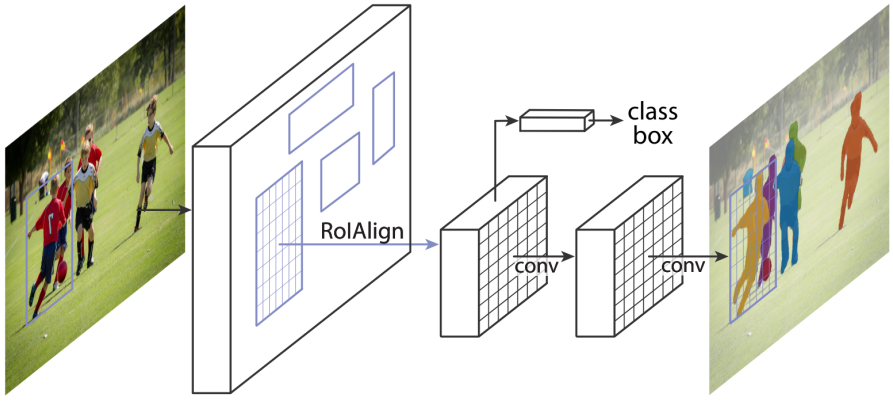


Figure 2.1: Mask R-CNN framework for instance segmentation [12]

RoI Align: A critical innovation in Mask R-CNN is the RoI Align approach, ensuring precise alignment of extracted features with input pixels. This technique accurately preserves the spatial positions of RoIs during pooling, thereby enhancing the accuracy of segmentation masks. The study highlights that RoI

Align effectively mitigates misalignment issues caused by quantization, thereby preserving spatial precision.

Segmentation Masks: Mask R-CNN generates a binary mask for each Region of Interest (RoI), enabling precise delineation of object boundaries at the pixel level. This is achieved through a mask head that produces separate binary masks for each class, ensuring accurate segmentation.

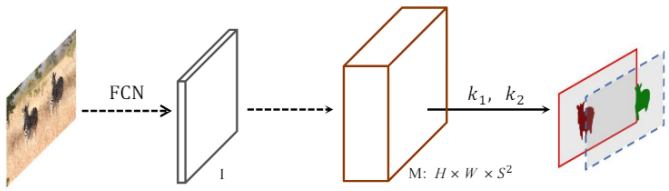
In material science, Mask R-CNN has been employed to detect and segment microstructural features in alloy materials, quantify defects, and analyze grain boundaries. The precise segmentation capability of Mask R-CNN is invaluable for conducting comprehensive microstructure analysis, allowing for the extraction of relevant information from complex images [13].

2.5 SOLOv2 model and comparison with Mask R-CNN

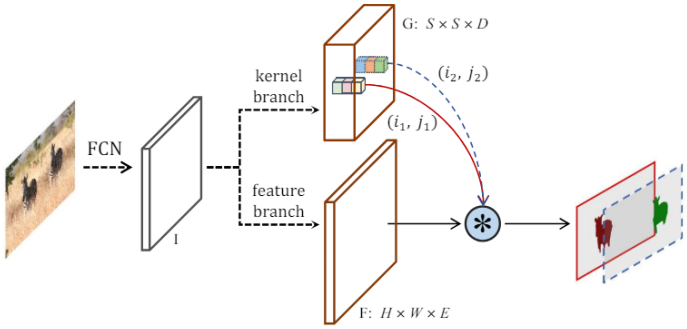
SOLOv2 represents a cutting-edge instance segmentation model that streamlines the process by treating instance segmentation as a direct prediction of object locations. Introduced by Wang et al. [14], this model introduces a straightforward and highly efficient approach to instance segmentation. Research on SOLOv2 underscores its innovative strategy of framing instance segmentation as a location prediction task, achieving notable improvements in both efficiency and accuracy compared to traditional methods. This novel approach enhances the effectiveness of instance segmentation by directly predicting the spatial extents of objects within images. Similarly to Mask-RCNN, here follows a breakdown of SOLOv2's structure:

Direct Prediction: SOLOv2 predicts object positions directly, eliminating the need for Region of Interest (RoI) operations and simplifying the segmentation process. This approach facilitates streamlined training and inference, enhancing speed and ease of use. Wang et al. [14] emphasize that SOLOv2 simplifies segmentation by framing it as a direct location prediction problem, thereby removing the complexity associated with RoI operations.

Efficiency: SOLOv2 is designed for computational efficiency, making it ideal for real-time applications and processing large-scale datasets. The model leverages dynamic convolutional kernels and a unified mask feature representation to deliver high performance while keeping computational overhead to a minimum.



(a) SOLO



(b) SOLOv2

Figure 2.2: Comparison of SOLOv2 to SOLO Architecture[14]

SOLOv2's efficiency and direct prediction capabilities make it particularly suitable for material science applications requiring rapid processing of large datasets. It has been utilized in projects involving real-time monitoring of construction progress, automated defect identification, and segmentation of electron microscopy images. Research by Wei et al. [15] demonstrated the effectiveness of SOLOv2 in automated segmentation tasks within material science, highlighting its reliability for analyzing extensive image datasets.

Both Mask R-CNN and SOLOv2 represent state-of-the-art advancements in instance segmentation, pushing the boundaries of accuracy, speed, and versatility in applications. These models have played a crucial role in advancing material science by providing accurate and detailed segmentation of microstructural features.

Precision vs. Speed: Mask R-CNN is renowned for its exceptional precision, largely due to its meticulous RoI Align and mask generation processes. However, this advantage comes with increased computational complexity. In contrast, SOLOv2 delivers faster processing times through its streamlined and direct

prediction methodology. Mishra et al. [11] demonstrated that Mask R-CNN excels in precision, whereas SOLOv2 offers a more efficient solution for real-time applications. The primary publication on SOLOv2 explicitly states that it achieves higher performance than its predecessor SOLO, with a 1.9% increase in average precision (AP) while being 33% faster [14].

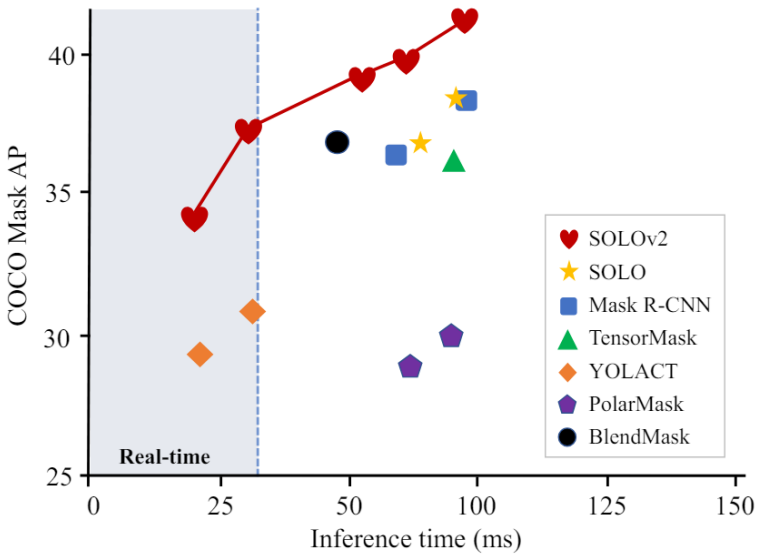


Figure 2.3: Speed vs. Accuracy with COCO test-dev. [14]

Complexity of Implementation: Implementing and fine-tuning Mask R-CNN is more intricate and resource-intensive compared to SOLOv2. The straightforward design of SOLOv2 simplifies its implementation and customization for diverse applications. Che et al. [13] concluded that the direct approach of SOLOv2 facilitates easier implementation, making it versatile for a wide range of applications.

Choosing between Mask R-CNN and SOLOv2 depends on the specific requirements of the task at hand. Mask R-CNN is preferred for applications demanding high precision and meticulous segmentation. On the other hand, SOLOv2 offers a more efficient solution for tasks requiring rapid processing of large datasets. Both models have significantly enhanced the ability to analyze

and understand complex microstructures in material science, leading to more accurate and comprehensive characterizations. According to Lin et al. [9], integrating these advanced models into material science has opened new avenues for in-depth research on microstructure, thereby enhancing the precision and effectiveness of material characterization.

2.6 Pertinent Research and Discoveries

Deep learning techniques have significantly advanced the analysis of materials science data, particularly within electron microscopy. The main aim of these studies has been to automate the detection and characterization of microstructural features like cavities, particles, and defects, which are essential for understanding material properties and behavior. This section provides an overview of key research projects and findings that highlight the effectiveness of deep learning models, such as Mask R-CNN and SOLOv2, in materials science.

Mask R-CNN is widely used in materials science because it can perform instance segmentation, which is crucial for detecting and quantifying microstructural features in electron microscopy images. The research applied the Mask R-CNN model to identify and measure nanoscale cavities in irradiated metals. The model demonstrated outstanding accuracy in providing swelling measurements for both individual images and specific conditions, which is vital for understanding how irradiation affects alloy performance.

Cohn et al. [16] conducted a detailed case study that specifically applied Mask R-CNN for microstructural analysis in materials science. In this study, Mask R-CNN was used to analyze images of metal powder particles produced by gas atomization. To handle the limited training set of labeled images, transfer learning was employed, enhancing the model's ability to generalize from a small amount of data. The Mask R-CNN model successfully generated comprehensive data on particle size distribution and the presence of satellite particles. This study demonstrated not only the adaptability and precision of Mask R-CNN in analyzing microstructural features but also its potential for broader applications in the field of materials science. The findings highlight the model's ability to provide valuable insights into the characteristics of metal powders, which are crucial for various industrial processes and material performance evaluations.

SOLOv2, a recent advancement in instance segmentation technology, has

been employed to address a variety of complex segmentation challenges in materials research. In a study conducted by Yang et al. [17], SOLOv2 was enhanced by integrating additional modules such as Feature Pyramid Grids (FPGs) and Convolutional Block Attention Modules (CBAMs). These enhancements significantly improved the model's feature extraction capabilities and segmentation speed. As a result, the study reported substantial improvements in both accuracy and efficiency of the segmentation process. This advanced technology holds great potential for applications in materials science, particularly for segmenting microstructural characteristics under various conditions. By utilizing SOLOv2, researchers can achieve more precise and rapid analysis of microstructures, aiding in the development and optimization of materials with desired properties for a wide range of industrial and scientific applications.

Comparative research has shown that deep learning models like Mask R-CNN and SOLOv2 outperform traditional image analysis algorithms. In a study by Dang et al. [18], the DeepLabV3+ model with a ResNet-152 backbone was compared to other advanced segmentation models. The findings highlighted the model's exceptional ability to detect and characterize various types of defects, underscoring the potential of these advanced models in materials science research.

The adoption of these advanced deep learning models has profoundly impacted materials science research. Automated segmentation and analysis have significantly reduced the time and effort required for manual annotation, allowing researchers to process larger datasets and obtain more accurate statistical results. This advancement facilitates a deeper understanding of material properties and behaviors, leading to more informed decisions in material design and development [19].

Despite the success of deep learning models in materials science, several challenges remain. These include the need for large, well-labeled datasets, the integration of specialized knowledge into the models, and the development of methods to handle complex and diverse microstructures. Future research should focus on addressing these issues, enhancing the models' applicability to various scenarios, and improving the interpretability of their results to advance the field further.

3

Methods & Implementation

This section outlines the specific approaches used in this research to build and utilize the Mask R-CNN and SOLOv2 models for measuring swelling in nuclear reactor alloys using electron microscopy images. The procedure starts by preparing the data, which involves converting JSON annotations into PNG masks and applying data augmentation techniques to improve the resilience of these models. Next, provided a detailed explanation of the implementation details for both Mask R-CNN and SOLOv2. Here emphasized the integration methods used to merge the outputs of both models in order to enhance the accuracy of segmentation. In addition, investigated the creation of customized loss functions that are specifically designed to include swelling signs, as well as the adjustments made to regular loss functions to better align with this specific application. Ultimately, constructed the guidelines for equitable comparison, elucidating the training and testing procedures to guarantee impartial assessment of the models' performance. This rigorous methodology guarantees that these results are strong, can be replicated, and offer useful insights into the implementation of deep learning in materials science.

Toolboxes and System Requirements: The study employs MATLAB R2024a to implement and execute the segmentation models. The following toolboxes are utilized:

Mask R-CNN:

- Computer Vision Toolbox: it provides essential functionalities for training of the Mask R-CNN. These include tools like Region of Interest (RoI) Align, which ensures precise mask generation, and pre-trained ResNet-based backbones that facilitate feature extraction.
- Deep Learning Toolbox: facilitates the training of deep neural networks, it offers opportunities to create custom layers, define loss functions, and manage the training process effectively.
- Computer Vision Toolbox Model for Mask R-CNN Instance Segmentation: this toolbox includes pre-trained models and specialized functions designed for instance segmentation using Mask R-CNN. It streamlines the process of training and deploying Mask R-CNN models.
- Compatible GPU: it is recommended to use an NVIDIA CUDA-enabled GPU to take use of GPU support, which enables faster computations and trainings.
- Parallel Computing Toolbox: it enables anyone to accelerate their computations by exploiting the power of multicore processors, GPUs, and computer clusters. It simplifies the implementation of parallel algorithms, allowing for faster execution of tasks such as simulations, data analysis, and optimization.

Besides this, few experiments, although not with the same degree of depth as in Mask R-CNN, will be conducted using SOLOv2. SOLOv2 requires these packages and toolboxes:

- Computer Vision Toolbox
- Deep Learning Toolbox
- Computer Vision Toolbox Model for SOLOv2 Instance Segmentation: basically the same as for Mask R-CNN.
- Compatible GPU
- Parallel Computing Toolbox

The toolboxes and system configurations are critical elements of this project, ensuring the effective training and execution of our models. Using a GPU significantly accelerates the training processes, which is vital given the complexity

and size of our datasets. Lastly, DeepLabV3+ for background segmentation basically uses the same toolboxes plus *Deep Learning Toolbox Model for ResNet-18 Network* (or alternatively ResNet-50 and ResNet-101), since the DeepLabV3+ model's backbone is a residual network.

3.1 Data Preparation

Dataset

The dataset used in this study is made up of electron microscope images of irradiated metal alloys, specifically chosen to detect and measure microscopic cavities. It is sourced from two primary locations:

- **1. Canadian Nuclear Laboratory (CNL):** This dataset features bright-field TEM micrographs of Inconel X-750 Ni alloys that have undergone neutron irradiation. After filtering for annotation accuracy, the CNL dataset includes 238 images.
- **2. Nuclear Oriented Materials & Examination (NOME) Laboratory at the University of Michigan:** This dataset contains 162 images of various steel alloys, such as CW-316, T91, HT9, and 800H, which have been exposed to both light and heavy-ion irradiation.

The merged dataset offers a wide variety of images that capture various irradiation settings and alloy compositions. This dataset provides a strong basis for training and assessing the segmentation algorithms. Displayed below are some images from the collection, illustrating the variations in cavity sizes, densities, and physical appearances across different imaging conditions:

Source	Material	Irradiation Type	Number of Images	Notes
CNL	Inconel X-750 Ni alloys	Neutron irradiation	238	Filtered for annotation accuracy
NOME	CW-316, T91, HT9, 800H steel alloys	Light and heavy-ion irradiation	162	Diverse alloy compositions

Table 3.1: CNL and NOME dataset's description

3.1.1 JSON to PNG Conversion for Mask Generation

The annotations of the project's dataset are saved in JSON format, providing detailed information about the segmentation masks of different cavities found in nuclear reactor alloys. Nevertheless, numerous deep learning frameworks

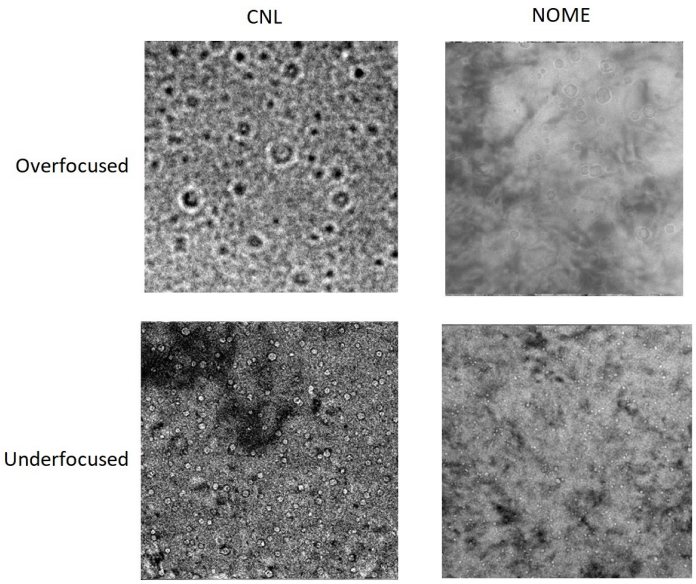


Figure 3.1: CNL and NOME focused and underfocused sample visuals

demand input in the format of images rather than annotations based on coordinates. Hence, the conversion of these JSON annotations into PNG images is an essential preprocessing step. In order to produce the desired segmentation masks as seen in figure 3.3, there are several mandatory steps to follow:

- **Reading JSON files:** The JSON files containing the annotations are processed and analyzed to extract the appropriate segmentation data. This include specific information such as the sizes of the images, the coordinates of the rectangular bounding box (expressed in the [x y w h] format), and the points that define the segmentation of each annotated cavity, providing the exact pixel-per-pixel classification inside the bounding box. This is accomplished by utilizing a script that loads the JSON file and decodes its content into a *struct* data structure that can be read by Matlab.
- **Setting Up Output Directories:** A suitable directory hierarchy is created to hold the PNG mask images that are generated. The output directory of the generated instance masks is explicitly defined, verifying the existence of the directory, and generating it if needed. The output directory is defined in relation to the current working directory, and the paths are modified accordingly.
- **Iterating Through Annotations:** The script sequentially examines the annotations of each image. The dimensions of the matching mask are extracted for each image, and an initial blank mask is constructed with all values set to zero (black color). The blank mask is sized according to the source picture to guarantee precise positioning of the cavities.

- **Extracting and Converting Segmentation Data:** The segmentation data, comprising a series of x and y coordinates that define the boundaries of the cavity, is processed for each annotation in the image. The provided coordinates are utilized to accurately describe the relevant cavity on the empty mask. The coordinates in the segmentation data are pairs of consecutive x and y numbers, indicating the pixel position of the boundary points of the instance.

- **Generating Masks:** The extracted coordinates are utilized to construct a polygon on the blank mask using Matlab's image processing algorithms. The polygon serves as a representation of the cavity, and the inside of the polygon is filled with the white color to generate a binary mask. The value within the polygon is assigned as 255, showing the existence of a cavity, while the background stays 0 (representing black).

- **Handling Edge Cases:** Errors or noise in the annotation data may cause certain segmentation points to fall beyond the valid range during mask construction. These locations are detected and removed to make sure the mask is generated with precision. If edge cases aren't handled properly, the resulting mask might get ruined due to an error in the segmentation data content. In case of bad coordinates for bounding boxes, the bounding box is directly thrown away and not considered at all.

- **Saving the Generated Masks:** Every mask that is created is stored as a PNG file. The naming convention incorporates the initial picture name and a numerical counter to distinguish between various cavities within the same image. This guarantees that each mask is distinctly recognizable and linked to the accurate image and cavity. For example, an image named **01_01.png** has masks named **01_01_0001.png**, **01_01_0002.png**, etc. as long as there are annotated cavities for such image.

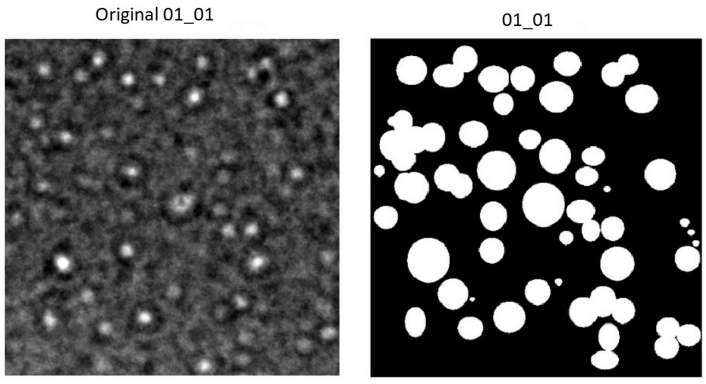


Figure 3.2: Original image and Ground Truth image with all instances in white

The process of converting JSON to PNG enables to utilize the aforementioned deep learning techniques and to finally set up the code to run the training of

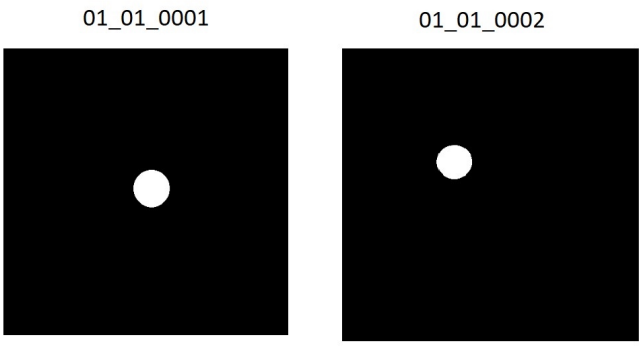


Figure 3.3: Examples of Ground Truth instance segmentation masks

the models. This approach guarantees the precise creation of masks, maintaining the spatial integrity of the annotations. Few mistakes were found in the provided annotations, so it is good practice to always take into account for potential mistakes and to find workarounds in those cases. The step of creating all instance masks requires a considerable amount of time (1-2 hours), because reading all data and creating new images when the dataset is big necessarily time consuming. Hence, this is done before launching a training and all instance masks are saved in the local disk and loaded at training run-time (not computationally expensive).

3.1.2 Data Augmentation and Preprocessing

Data augmentation is a technique used to increase the diversity and quantity of training data without actually collecting new data. This is achieved by applying various transformations to the existing dataset, such as rotations, translations, scaling, flipping, and color adjustments. By introducing these variations, the models are exposed to a wider range of possible scenarios during training. As far as the CNL and NOME datasets are concerned, the presence of round-ish instances lead to believe that only translations (shifting the image by few pixels) and scaling factors (increasing or decreasing the image sizes) were useful, however it is not clear whether it is required, given the large amount of instances available. On the other hand, preprocessing was a much more needed step to prepare the data for using and training the models considered; the pre-

processing of data ensures that the data is in the appropriate and consistent format for the model to use. Another vital step that will be explored in this section is the *stacking* of instance masks. As requested by the Mask R-CNN and SOLOv2 frameworks, for each training or validation image there must be a one to one correspondence with the entire set of instance masks of that image. This means that there is another preprocessing regarding how the data is handled. First of all, a new matlab data file (.mat format) is created, to ensure consistency between names, this file is named after the image it represents. Secondly, the .mat file contains a 3D logical array, where the first two dimensions represent the sizes of the image, and the third dimension represents the amount of binary masks to be associated with the image. What follows is the sequence of multiple preprocessing steps required to run the Mask R-CNN or SOLOv2 models:

- **Loading Training Data:** the first step is to import the training images into Matlab via the creation of an image datastore. This datastore contains the explicit path of all training images of our dataset, and it loads them in a format which is appropriate for processing. By default, the training images are saved in a directory called *train* inside the dataset folder.
- **Handling Bounding Boxes:** bounding boxes are rectangular outlines used to identify the position and size of objects within images. The annotation file contains bounding boxes for each cavity, for each image. These bounding boxes are stored in a cell array, with each cell corresponding to an image and containing its respective bounding boxes. The number of bounding boxes varies for each image, depending on the number of cavities present. For example, one image might have 100 bounding boxes, while another could have 150, as detailed in the table format.
- **Categorical Labels:** categorical labels, usually in text format, are used to classify objects within bounding boxes. In this case, each bounding box is associated with the label "cavity," indicating the object's category. These labels are stored in a cell array that mirrors the structure of the bounding box cell array. Each entry in the label array directly corresponds to an entry in the bounding box array, in a one-to-one relationship between labels and bounding boxes.
- **Combining Bounding Boxes and Labels:** bounding boxes and labels are combined into a table, which associates each bounding box with its corresponding label to ensure precise identification of each cavity. This table is essential for training the model, as it provides the required structured framework that enables the model to learn the relationship between image regions and object categories.
- **Loading Instance Masks:** Instance masks are binary masks that precisely depict the shape and position of individual objects inside an image. The instance masks, as mentioned before, are saved as .mat files, and they

are loaded using a custom function for reading datastores. This method parses the .mat files and extracts the binary masks. Each .mat file provides a collection of instance masks for a single image. The custom read function is required because the usual datastore reader lacks the capability to directly process the .mat file format.

- **Combining Datastores:** a combined datastore is a matlab varibale type that includes several datastores, in order to ease how data is handled. In this case, the combined datastore contains the image datastore for the training images, the datastore made out of the table containing labels plus categories and finally the ground truth stacked masks in the .mat files. The integrated datastore contains all the essential data required for training, guaranteeing that each training image is linked with its corresponding bounding boxes, labels, and instance masks. This integration guarantees that the model has full access to all pertinent data during the training process.

- **Data Augmentation:** in this optional step, training images, along with their related masks and bounding boxes, undergo data augmentation techniques as explained earlier. These augmentations enhance the variety of the training set, improving the model's ability to generalize with unfamiliar data. Needless to say, augmentation is implemented so that correspondences between images and their annotations are preserved.

- **Normalization and Preprocessing:** The images have been normalized to a common range, to guarantee consistent input for the deep learning models. To ensure uniformity throughout the dataset, any extra preprocessing processes, such as resizing images and masks to a uniform size, are carried out. Normalization helps stabilize the training process and ensures that input data is properly formatted for the neural network. By following these procedures, data is configured optimally for training deep learning models. Augmentation techniques enhance model robustness, and preprocessing steps normalize the input data, facilitating effective training and accurate predictions. A comprehensive data preparation strategy is crucial for achieving the best results in segmentation tasks.

3.2 Implementation of Mask R-CNN

Now that the preprocessing steps were carefully explained, let's dive into the implementation of the Mask R-CNN model. This section outlines the entire process, from data preparation to training options, incorporating custom scripts and methods written to conform with the project's specific requirements.

3.2.1 Data Preparation:

1. Annotation Processing:

The JSON format annotations are processed using proprietary MATLAB scripts, readJson.m and the optional readJsonVal.m, in case validation data needs to be used. The scripts in question are responsible for loading the JSON files and decoding their contents into structures that can be read by Matlab. Again, these structures contain information about image dimensions, bounding box coordinates, labels and segmentation points for each annotated cavity. Since all instance masks are stored locally, only the bounding boxes information and labels are used.

2. Datastore creation:

- **Image Datastore (imds):** this datastore contains the training images that are saved in the "train" folder. It is initialized to simplify the loading and processing of the images.
- **Box Label Datastore (blds):** this datastore holds the bounding box coordinates and corresponding labels, where each bounding box is labeled as 'cavity'. The bounding boxes are organized in a cell array, with each cell corresponding to an image.
- **Instance Mask Datastore (imdsInstances):** this datastore reads the binary masks contained in the .mat files. A custom read function dsImageReader.m is used to load these masks correctly and to make the datastore readable at training time.

3. Combining Datastores:

The datastores are merged into a unified datastore (cmbds) using the combine function. The combined datastore guarantees the accurate association of each

training image with its corresponding bounding boxes, labels, and instance masks. This facilitates efficient data loading throughout the training process.

3.2.2 Network Initialization:

Mask R-CNN is based on the ResNet-50 structure, and then trained on the widely known COCO (Common Objects in COntext) dataset, containing a handful of, as the name suggests, everyday objects. This means that the base model is defined upon this dataset, and not our custom dataset, implying the necessity of modifying the underlying structure of the neural network. The results of the modification of such parameters will be explained in chapter 4.

1. Custom Anchor Boxes:

There are several way to obtain anchor boxes. One can hand-craft the anchor boxes and use those, otherwise one can use a k-means clustering algorithm to generate the anchor boxes that best fit the data at hand. The original base paper [1] used 7 squared anchor boxes of size [4, 8, 16, 32, 64, 128, 256].

2. Network Architecture:

The Mask R-CNN model is constructed using a ResNet-50 backbone that has been pre-trained on the COCO dataset. The backbone of this system extracts features from the input images, which are subsequently processed by the following components:

- **Region Proposal Network (RPN):** Generates region proposals from the extracted features.
- **Detection Head:** Performs bounding box regression and classification.
- **Mask Head:** Predicts segmentation masks for each region of interest.

With the current Matlab tools and availability, only a ResNet-50 backbone is available.

Transfer Learning:

Transfer learning utilized to customize the pre-trained ResNet-50 backbone for the project's particular dataset consisting of nuclear reactor alloy cavities. This process entails optimizing the network by training the feature extraction

layers again using the dataset at hand. Transfer learning enhances the model's accuracy and performance on specialized task by enabling it to acquire the specific features of cavities, since, obviously, the task does not contain the same categories as in the COCO dataset. Through the process of re-training the feature extraction layers, the model is capable of modifying its weights and biases in order to improve its ability to accurately detect and segment the cavities that are visible in the alloy images. The process of fine-tuning is essential in order to customize the general pre-trained model to the individual application.

3.2.3 Training Configuration

Training Parameters:

- Initial Learning Rate: set to 0.005 to begin the adjustments to our data.
- Learning Rate Schedule: to refine learning, every 10 epochs the learning rate decays by a factor of 0.95.
- Momentum: set to 0.9 for stable convergence.
- Max Epochs: several hundreds, this however will be better explained in chapter 4.
- Mini-Batch Size: 4, using as much GPU memory as possible and to obtain the great accuracy in output. Usually this does not imply a faster convergence, but a better performance in the output model.
- Execution Environment: configured to utilize a GPU for faster computations. In the provided Blade Cluster, an NVIDIA A40 graphics card was used to satisfy the huge memory demands of the batch size and the 512x512 training image sizes.

Training Procedure:

- The network is trained using Stochastic Gradient Descent with Momentum (SGDM). The training loop monitors the loss and adjusts the learning rate as per the predefined schedule.
- Early Stopping: optional, not used, it is implemented to prevent overfitting. Training halts if the validation loss does not improve for a specified number of epochs.
- Validation Checkpoints: Regular evaluation on the validation set to obtain the best-performing model, based on validation metrics.

3.3 Implementation of SOLOv2

The Matlab framework to build a SOLOv2 model essentially follows the same preprocessing steps described in the Mask R-CNN implementation section. For this reason, there won't be a repetition of these steps. A SOLOv2 model was built as a research for better results, however it was not dealt with to the same degree as for the Mask R-CNN model. It is however important to mention the capabilities and functionalities of SOLOv2, for eventual future implementations.

3.3.1 Network Initialization:

1. Grid Cell Assignment:

SOLOv2 assigns objects to grid cells on feature maps, where each grid cell predicts the presence of an object and generates a mask for that object. Unlike traditional R-CNN-based models, SOLOv2 does not use anchor boxes. Instead, it utilizes direct grid cell assignments for detecting and segmenting objects.

2. Network Architecture:

- **Backbone Network:** just like Mask R-CNN, the SOLOv2 model uses a ResNet-50 backbone for feature extraction, pre-trained on the COCO dataset. This backbone is responsible for extracting rich feature maps from the input images.
- **Feature Pyramid Network (FPN):** the FPN enhances feature maps at multiple scales, improving the detection and segmentation of objects at various sizes.
- **Mask Kernels and Features:** SOLOv2 divides the mask prediction into two branches: mask kernels and mask features. The mask kernels dynamically create masks for each object, while the mask features offer detailed spatial information for an accurate segmentation.

Transfer Learning:

Following the same reasoning for the Mask R-CNN model, transfer learning is used to learn how to extract features from the CNL/NOME datasets.

3.3.2 Training Configuration

Training Parameters:

- Initial Learning Rate: set to 0.01, suitable for the SOLOv2 architecture according to the Matlab documentation
- Learning Rate Schedule: no clear rule was used here, therefore the same settings as for Mask R-CNN were used
- Momentum: set to 0.9..
- Max Epochs: 50, providing sufficient training time for an output, although the amount of epochs might need to be reconsidered.
- Mini-Batch Size: 4 or 8, balancing memory usage and training efficiency.
- Execution Environment: configured to utilize a GPU for faster computations. In this case, the home laptop was utilized, using an RTX 3060 (laptop version, slower than the correspondent PC version).

Training Procedure:

- The network is trained using Stochastic Gradient Descent with Momentum (SGDM). The training loop monitors the loss and adjusts the learning rate periodically based on the predefined schedule.
- Early Stopping: implemented to prevent overfitting. Training halts if the validation loss does not improve after few consecutive epochs. In this case early stopping was implemented to obtain a "quicker" model, not necessarily implying it is the best model.
- Validation Checkpoints: the model is evaluated on the validation set at regular intervals, and the best-performing model is saved.

Since SOLOv2 uses essentially the same preprocessing procedures for a Matlab implementation, it became a trivial task to implement it for instance segmentation. That is the underlying reason as to why it was implemented, offering a comparison of results with Mask R-CNN. This is thanks to Matlab's easily accessible documentation and interface, and cohesion between models, despite SOLOv2 architecture being quite different compared to Mask R-CNN's architecture.

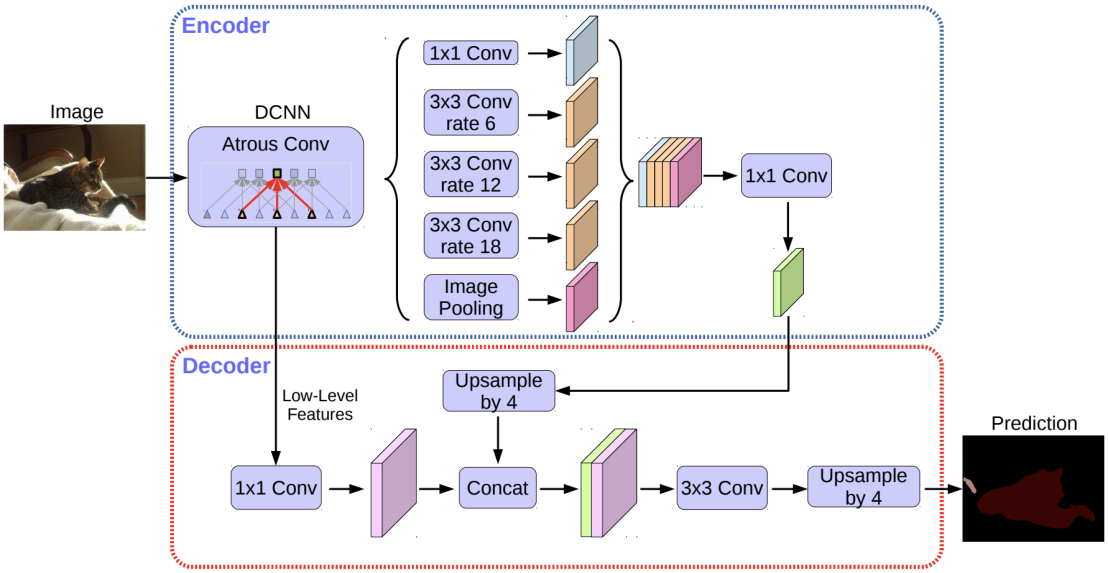


Figure 3.4: DeepLabV3+ architecture

3.4 DeepLabV3+

DeepLabV3+ (figure 3.4) is a state-of-the-art deep learning model designed for semantic image segmentation tasks, which aims to label each pixel in an image with a corresponding class. In this thesis, the purpose of implementing a DeepLabV3+ model is to attempt to obtain a background segmentation to be exploited by Mask R-CNN in some ways to be defined. By segmenting the training images in background-cavity classes, the hope is to extract extra information to run the training from. This is the framework to build a functional DeepLabV3+ model in Matlab, after having implemented a Mask R-CNN:

- Have the appropriate toolboxes, they're essentially the same ones required by Mask R-CNN and SOLOv2, plus the extra *Deep Learning Toolbox Model for ResNet-50 Network* toolbox, since DeepLabV3+'s backbone is a Residual Network.
- This model utilizes binary masks, not instance masks. Therefore, the ground truth images must be built by combining all instance masks related to the same image, in order to create a black and white image where each circle represents a cavity, as shown in figure 3.5. These masks are built before training the model.
- Two datastores are initialized. One that contains the training images, the second one that contains the ground truth binary masks and that holds the pixel information (white=cavity, black=background). Data augmentation is applied in the form of translation.

- Without spending too much time on training parameters, standard training option parameters, available in Matlab's documentation, were used.

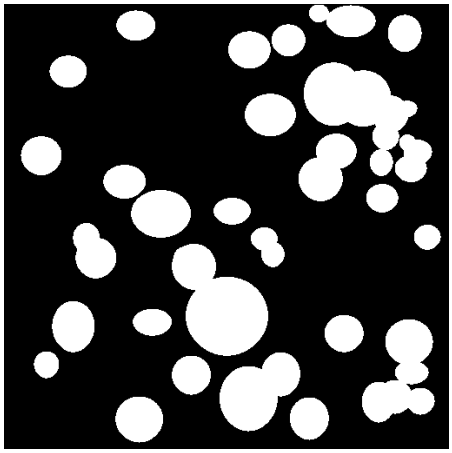


Figure 3.5: Binary ground truth mask used for DeepLabV3+

3.5 Mask R-CNN and SOLOv2 inference time

Once both models were trained on the dataset, it was required to build a way to evaluate and potentially compare the results obtained. In order to also have a fair comparison with the baseline paper's results, the same metrics were implemented: precision, accuracy and f1 scores; ground truth number of cavities, number of predicted cavities and number of cavities found; true density of cavities within an image and the corresponding predictions; true size of cavities versus predicted size of cavities; finally, ground truth swelling of an image and predicted swelling.

Again, the implementation of the inference time is relatively simple given Matlab's documentation. To run the models on the validation set, all it took was to prepare the validation set data the same way used for the training data, that is, to load the data into the required datastores. Using the *segmentObjects* function, a new datastore and folder of stacked masks in the form of .mat files are created. Then, a non-maximum suppression (NMS) algorithm is run on those stacked masks, removing potential overlapping masks. Infact, it can happen that a small instance is detected within an instance, which doesn't make sense in this application.

The final step that retrieves precision, accuracy and f1 scores is to simply use the *evaluateInstanceSegmentation* function, ran on the datastore that contains the stacked masks after applying NMS.

4

Experiments and Analysis

4.1 Finding Mask R-CNN parameters

This section is dedicated to all experiments and parameter tuning to attempt to find a way to improve the accuracy of the Mask R-CNN model. In fact, after having set up the data correctly, the natural question that rose to mind was *"What values should be assigned to the various parameters?"*

A lot of testing was done in the first place, mostly due to the difficulties with actually finding those parameters and understanding what changes they produced when edited. This is usually one of the most challenging aspects in deep learning, since there is no guide that tells you how to set your parameters, rather, you have to search for them.

In the following subparagraphs there will be a discussion about the reasons why certain parameters were changed, in a more or less chronological order.

4.1.1 Default Mask R-CNN

The default version of the model comes with fixed anchor boxes sizes, and it is pretrained on the COCO dataset. The training loss, a metric used while training a model to establish how well the model is performing on the training set, was always performing badly. This naturally lead to believe that maybe longer a longer was required, however this hypothesis turned out to not be true, because no matter the learning rate, batch size the training loss was somewhat flat at a relatively high number. Moreover, the output of the model when given

the test set was essentially empty, indicating that the model didn't learn.

4.1.2 Anchor boxes linear estimation

Naturally, the first parameter that came to mind to tweak was the anchor boxes. As a reminder, the anchor boxes define the windows that capture the regions of interests that contain instances/objects to be found. Therefore it was assumed that the default version of Mask R-CNN was not able to find enough instances to even train efficiently.

Hence the conclusion that the dataset required better anchor boxes, because there was a need for anchor boxes that were representative for the data. Without going into a lot of details, a linear model for estimating the anchor boxes was made following this procedure: obtain all ground truth cavity sizes and plot them in a width-height graph, in the x and y axes, then linearly fit some square anchor boxes to make sure that they can capture the cavities.

Few amount of anchor boxes were attempted, 8, 16 and 24. Using more than 24 anchor boxes leads to unnecessary slowdowns in the RoI selection at training time, since more anchor boxes implies a bigger search for instances. No results were obtained and this procedure was discarded completely.

4.1.3 Anchor boxes estimation using k-means clustering

"Well, maybe using a linear estimation isn't enough to be representative of the data." So a k-means clustering algorithm was implemented quickly, thanks again to Matlab documentation and the Statistics and Machine Learning toolbox. In the figure 4.1, a plot is shown to better understand how the anchor boxes work. Essentially each anchor box should be dedicated to capture as many instances as possible around its centroid.

Despite trying 9, 16 and 25 anchor boxes, none of them worked, therefore this method was also discarded, even though several training options were tried as always.

4.1.4 Subset of training images

Another idea was that perhaps it wasn't because of the training options but because the model simply needed more time to train. The idea behind this strategy was to select few images (about 30% of the training set), without a

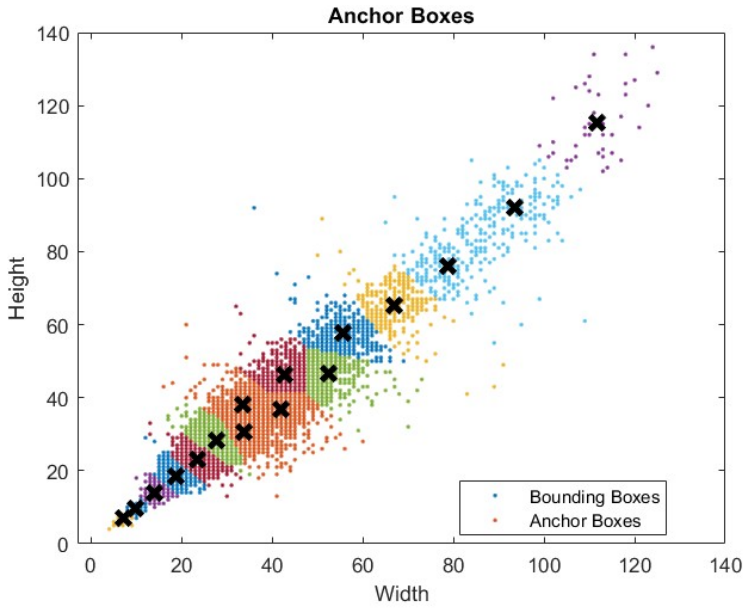


Figure 4.1: Plot of 16-means clustering to fit the ground truth masks in order to spot instances for Mask R-CNN

specific criterion, to overfit the model on this subset to potentially find the issue behind the high training loss. If the model actually had no problems learning how to detect instances, it would overfit more quickly.

So by still using k-means clustering, since it anyways gave the best performance thus far, more training routines were launched. The training loss did become smaller but the results were still disappointing, so the attention on improving the model shifted elsewhere.

4.1.5 Different network input sizes and different image subset

Another idea was that the network wasn't able to learn to detect enough instances because the resolution of the images wasn't satisfactory. The images were upscaled by 2x their size, along with adjusted annotations and anchor boxes sizes. Furthermore, another different image subset was used, using the criterion that the images shouldn't contain more than 100 instances.

However, despite these attempts, nothing really worked. It was also hypothesized that in the default Matlab implementation the model cannot detect more than 100 instances, which obviously might have affected the training. This theory will be explored more in detail in 4.3.

4.1.6 Best anchor boxes

In the original paper, they reported a given set of 7 anchor boxes, values that originally were not chosen to first experiment freely. However, since nothing worked, the set of square sizes [4; 8; 16; 32; 64; 128; 256] anchor boxes provided great results, despite requiring an enormous amount of training.

The overall training loss kept decreasing even after more than 200 epochs, so the final proposed model was trained on 500 epochs in the Blade Cluster, a training that took about 20 hours to complete, using a minibatch size of 4 and a decaying learning rate from 0.005 to 0.001.

Despite the long training, no overfitting was ever observed. This is probably due to the amount of instances in the training dataset, which contains over 20.000 cavities.

4.2 Evaluation metrics

When training a deep learning model, such as fine-tuning models like Mask R-CNN and SOLOv2, it is extremely important to evaluate their performance on a test set that the model has not encountered before during training time. The aim of this evaluation is to determine the model's ability to effectively generalize to new and unseen data. When the model is tested on the same dataset used for training or validation, the results might be overly optimistic. This occurs because the model might merely memorize patterns in the training data instead of genuinely learning to apply that knowledge to novel examples. Accuracy, a common metric in model evaluation, measures the proportion of correctly predicted instances out of the total number of observations. However, its usefulness can be limited with imbalanced datasets, where one class predominates. In addition to evaluating the model's performance, it is important to conduct a thorough analysis of other subsequent metrics [20].

1. Precision is the degree of accuracy or exactitude in measuring. In simple words, it is the ratio between the number of correctly found instances over the number of found instances and the mistakenly predicted positive instances. Achieving low precision might either mean that the model didn't find enough true labels or that it mistakenly detected too many labels to be positive. The metric relates to the accuracy of positive forecasts. The precision formula is

defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

2. Recall, also known as sensitivity or the true positive rate, is the proportion of correctly identified positive instances out of the total number of actual positive instances in a classification model. This metric evaluates the model's ability to detect and capture all relevant events. In other words, recall measures how well the model predicts positive observations from the total number of true positive instances. The formula for calculating recall in a binary classification problem is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

3. The F1 Score is a commonly used metric in machine learning and statistics to evaluate the performance of a classification model. It offers a balance between precision and recall, making it particularly useful in scenarios with class imbalance. It is also regarded as the harmonic sum between precision and recall, and often mentioned as the "F1 curve/plot". The formula for calculating the F1 score is as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the field of instance segmentation, as in other areas of deep learning, it is almost mandatory to employ metrics such as Precision, Recall, and F1 Score to evaluate a model's performance. These metrics provide insightful information regarding the model's ability to accurately detect relevant data while minimizing false positives and false negatives. When assessing a model on a test set, it is ideal to achieve high precision and recall, which subsequently results in a high F1 Score. The F1 Score, as just mentioned, represents a harmonious balance between precision and recall, offering a comprehensive measure of the model's effectiveness.

In addition to these metrics, the Confusion Matrix serves as an excellent tool for evaluating a model's performance. This matrix is particularly valuable for understanding the classification system's performance across multiple categories. It provides detailed insights into the specific types of errors the model makes, not just the overall error rate. The confusion matrix visually maps actual classes to rows and predicted classes to columns, breaking down the model's predictions into four categories: True Positives (correctly predicted positive instances), True Negatives (correctly predicted negative instances), False Positives

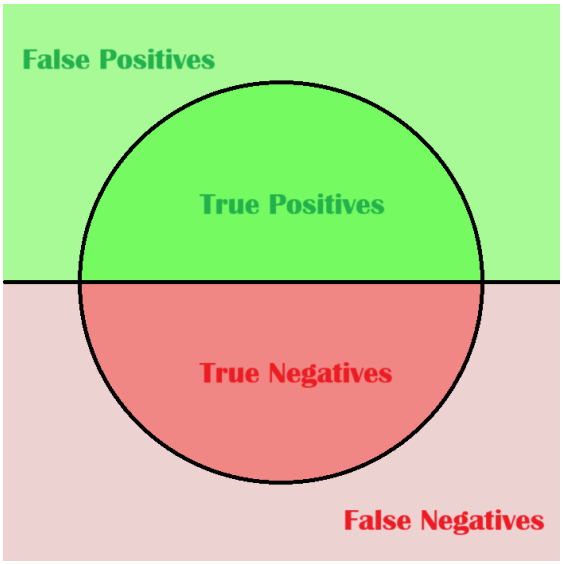


Figure 4.2: Visualization of True/False Positive and True/False Negative

(incorrectly predicted positive instances), and False Negatives (incorrectly predicted negative instances). This is visualized in figure 4.2.

Using the confusion matrix, one can derive Precision, Recall, and Accuracy values, which, along with the F1 Score, offer a detailed assessment of the model's performance. These metrics collectively ensure a thorough evaluation of the model's suitability for specific tasks in instance segmentation, highlighting both its strengths and areas for improvement. This comprehensive analysis is vital for developing models that are not only accurate but also reliable and efficient in real-world applications.

Additional metrics from the base paper [1] are listed and represented below:

4. Total True represents the total number of ground truth masks in the dataset. Assuming N ground truth masks for an image, the total true formula is defined as:

$$\text{Total True} = \sum_{i=1}^{N} \text{Ground Truth Masks}$$

5. Total Pred represents the total number of detected instances by the segmentation model. Assuming the model finds M instances, the total pred formula is defined as:

$$\text{Total Pred} = \sum_{i=1}^{M} \text{Predicted Positive Instances}$$

6. Total Found represents the total number of masks found, where a mask is considered found if there is an overlap between the predicted mask and a ground truth mask. Clearly, the number of found masks is less than or equal to the number of predicted masks. The total found formula is defined as:

$$\text{Total Found} = \text{True Positives}$$

7. True Density measures the actual density of cavities in the images. It is calculated as the ratio of true positive instances to the image area. There can be several indices for the image area. In this thesis, not knowing the exact dimensions of the S/TEM images, the image area in pixels was used. The total density formula is defined as:

$$\text{True Density} = \frac{\text{Total True Masks (from GT)}}{\text{Image Area}}$$

8. Pred Density measures the predicted density of cavities by the model. The pred density formula is defined as:

$$\text{Pred Density} = \frac{\text{Total Pred Masks}}{\text{Image Area}}$$

9. Density Error quantifies the error in the predicted density compared to the true density. The density error formula is defined as:

$$\text{Density Error} = \left(\frac{|\text{True Density} - \text{Pred Density}|}{\text{True Density}} \right) \times 100$$

where the error in density is reported as a percentage of our predicted density w.r.t. true density, in absolute value.

10. True Size measures the actual size of the cavities in the images. This is quantified by calculating the average size of all true positive instances. The true size formula is defined as:

$$\text{True Size} = \sum \text{Area of Ground Truth Masks}$$

Meaning that the true size of an image i is the sum of the areas of the GT masks in pixels. In the base paper, they report a per-cavity true size measure, however here the total area in pixels is reported.

11. Pred Size measures the predicted size of the cavities by the model. The pred size formula is defined as:

$$\text{Pred Size} = \sum \text{Area of Predicted Masks}$$

12. Size Error is the error in the predicted size compared to the true size. The size error formula is defined as:

$$\text{Size Error} = \left| \frac{\text{True Size} - \text{Predicted Size}}{\text{True Size}} \right|$$

13. True Swelling is the measure of the actual swelling in the material as determined from the ground truth masks. The swelling indicator basically estimates how swollen the material is, that is, how much the material protrudes from the 2D plane. The true swelling formula is defined as:

$$\text{True Swelling} = 100 \times \left(\frac{\frac{\pi}{6} \sum_{j=1}^{n} d_j^3}{\text{imageArea} - \frac{\pi}{6} \sum_{j=1}^{n} d_j^3} \right)$$

where n is the number of true masks and d_i is the i-th mask.

14. Pred Swelling is the predicted amount of swelling by our model. The formula is omitted since it's basically the same as the True Swelling formula.

15. Swelling Error measures the difference between the true swelling and the predicted swelling. It's calculated as the absolute difference between the true swelling and the predicted swelling over true swelling, in terms of percentage. The swelling error formula is defined as:

$$\text{Swelling Error} = 100 \times \left| \frac{\text{True Swelling} - \text{Predicted Swelling}}{\text{True Swelling}} \right|$$

Given all of these metrics, it is possible to estimate the model's performance to a great degree of comprehension, further understanding where the model is lacking and where the model shines, in this specific dataset.

4.3 Implementation Analysis and Results

Before evaluating the Mask R-CNN, it is important to note that a major change in the provided Matlab code was made. In fact, by running the *segmentObject*

function that finds all instances within images, it was noticed that the output always contained at maximum 100 instances. By doing some manual inspection, it was found that "100" is hardcoded in the Matlab library to be the maximum number of detectable objects in an image. It is not clear whether this is exactly the case, however after modifying this value within the Matlab installation files (the file is simply called MaskRCNN.m), to 800, assuming that no image contained more than 700 instances by inspecting the ground truth data, the model found all instances.

It is never recommended to edit installation files, however this was the only workaround that solved momentarily the issue.

It is important to note that this change was not made inside the Blade Cluster, therefore it is unknown whether the model predicted all instances at training time.

4.3.1 Mask R-CNN Evaluation

The general evaluation procedure works as follows:

- **Dataset Splitting:** The dataset was split into training and testing sets, ensuring that the testing set contains images not seen by the model during training to evaluate its generalization capabilities.

- **Model Inference:** The trained Mask R-CNN model segmented the test set images to generate predictions for bounding boxes, instance masks, and segmentation outputs.

- **Metric Calculation:** The evaluation script calculates the precision, recall, F1 score, density error, size error, and swelling error for each image in the test set. These metrics were computed by comparing the predicted outputs to the ground truth annotations.

- **Visualization and Analysis:** The results were visualized and analyzed to identify any patterns or areas of improvement. The confusion matrix in this case was not used due to the presence of just one class.

ImageIndex	Precision	Recall	F1Score	TotalTrue	TotalPred	TotalFound
1	0,4902	0,9259	0,6410	54	102	50
2	0,7159	1,0000	0,8344	63	88	63
3	0,5147	1,0000	0,6796	35	68	35
4	0,6311	0,9848	0,7692	66	103	65
5	0,7952	1,0000	0,8859	66	83	66
6	0,6962	1,0000	0,8209	55	79	55
7	0,2907	1,0000	0,4505	25	86	25
8	0,4512	0,9737	0,6167	38	82	37
9	0,6146	0,9833	0,7564	60	96	59
10	0,0563	0,3667	0,0976	60	391	22
11	0,4978	0,7019	0,5825	161	227	113
12	0,5708	0,7247	0,6386	178	226	129
13	0,5971	0,7455	0,6631	165	206	123
14	0,6000	0,8163	0,6916	147	200	120
15	0,0657	1,0000	0,1232	26	396	26
16	0,2093	0,4449	0,2847	263	559	117
17	0,9305	0,7915	0,8554	609	518	482
18	0,4061	0,4010	0,4035	399	394	160
19	0,3170	0,4551	0,3737	312	448	142

Figure 4.3: Evaluation of Mask-RCNN, standard metrics

4.3.2 Results for Mask R-CNN

The first table as seen in image[1] 4.3 displays the results for the Mask R-CNN model as far as the standard metrics are concerned. This includes precision, recall, F1 score and the true number of instances, the predicted and found number of instances.

The second table, image 4.4, instead contains the results as far as the custom metrics are concerned, namely the true, predicted densities and density error; true, predicted sizes and size error; true, predicted swellings and swelling errors.

Summary of Metrics

1. F1 Score:

- The F1 score ranges from 0.0976 *(Image 10)* to 0.8859 *(Image 05)*, indicating variability in the model's performance across different images.

[1]The author understands that it is ugly to report the results as images of excel sheets. Nevertheless, formatting the table in a space-efficient way was an impossible task, given tables of size 20x18.

img	TrueDensity	PredDensity	DensityError	TrueSize	PredSize	SizeError	TrueSwelling	PredSwelling	SwellingError
1	0,000206	0,000389	88,89	55737	72102	29,36	21,26	27,50	29,36
2	0,000240	0,000336	39,68	104122	101224	2,78	39,72	38,61	2,78
3	0,000134	0,000259	94,29	73098	102611	40,37	27,88	39,14	40,37
4	0,000252	0,000393	56,06	91647	93630	2,16	34,96	35,72	2,16
5	0,000252	0,000317	25,76	108473	103258	4,81	41,38	39,39	4,81
6	0,000210	0,000301	43,64	62503	74525	19,23	23,84	28,43	19,23
7	0,000095	0,000328	244,00	37573	86990	131,52	14,33	33,18	131,52
8	0,000145	0,000313	115,79	40782	76729	88,14	15,56	29,27	88,14
9	0,000229	0,000366	60,00	56625	75799	33,86	21,60	28,92	33,86
10	0,000229	0,001492	551,67	8076	22731	181,46	3,08	8,67	181,46
11	0,000614	0,000866	40,99	48382	30609	36,73	18,46	11,68	36,73
12	0,000679	0,000862	26,97	65610	40357	38,49	25,03	15,39	38,49
13	0,000629	0,000786	24,85	69083	42182	38,94	26,35	16,09	38,94
14	0,000561	0,000763	36,05	56706	62836	10,81	21,63	23,97	10,81
15	0,000099	0,001511	1423,08	8804	22600	156,70	3,36	8,62	156,70
16	0,001003	0,002132	112,55	18372	15595	15,12	7,01	5,95	15,12
17	0,002323	0,001976	14,94	39414	22337	43,33	15,04	8,52	43,33
18	0,001522	0,001503	1,25	27671	14425	47,87	10,56	5,50	47,87
19	0,001190	0,001709	43,59	19426	17952	7,59	7,41	6,85	7,59

Figure 4.4: Evaluation of Mask-RCNN, custom metrics

- The average F1 score indicates how well the model balances precision and recall.
- The average F1 score for the reported values is 0.5878, indicating an average score.

2. Precision and Recall:

- Precision values range from 0.0563 *(Image 10)* to 0.9305 *(Image 17)*, indicating the presence of a lot of false positives (since the corresponding F1 scores are by and large medium).
- Recall values range from 0.3667 *(Image 10)* to 1.00. For this application, it is thought that obtaining a high recall but a low precision is fine, since it is better to highlight the presence of more cavities rather than to miss the cavities entirely.
- High precision and recall for certain images indicate good detection accuracy, while lower values suggest missed or incorrect detections.

3. Density and Size Errors:

- True and predicted densities, along with their errors, provide insights into the model's ability to estimate the concentration of cavities in an image. Given the great amount of false positives, despite the NMS to eliminate part of them, the model presents several defects in the density estimation of instances.

- Size errors explain the difference between the predicted and true cavity area within each image, except few cases, the model has obtained alright results in guessing the amount of cavity area in a material.

4. Swelling error: Swelling errors indicate that the model is somewhat capable of estimating the swelling of an image, again, despite few bad cases. One might obtain a better swelling estimation if the a coefficient for the image area is utilized, instead of using pixel analysis.

Detailed Analysis

High Performance Image:

- *Image 05:* This image (figure 4.5) has the highest F1 score with high precision and recall. The density and size errors are relatively low, indicating accurate predictions.

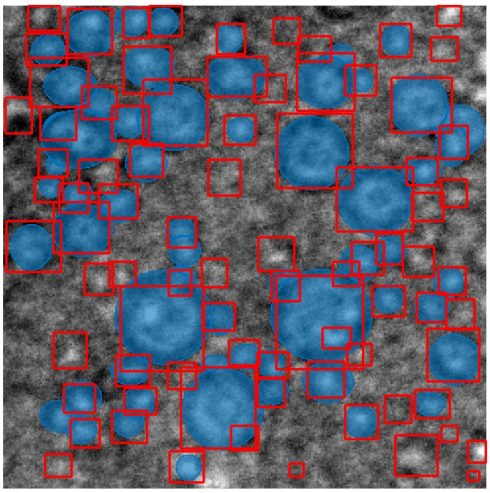


Figure 4.5: The blue spots indicate the ground truth cavities, while the red boxes indicate what the model perceived to be a cavity, with the segmented area omitted for visibility purposes

Low Performance Image:

- *Image 10:* This image shows the lowest F1 score (0.3246119) and precision (0.3037037). The high density and size errors indicate significant discrepancies between true and predicted values.

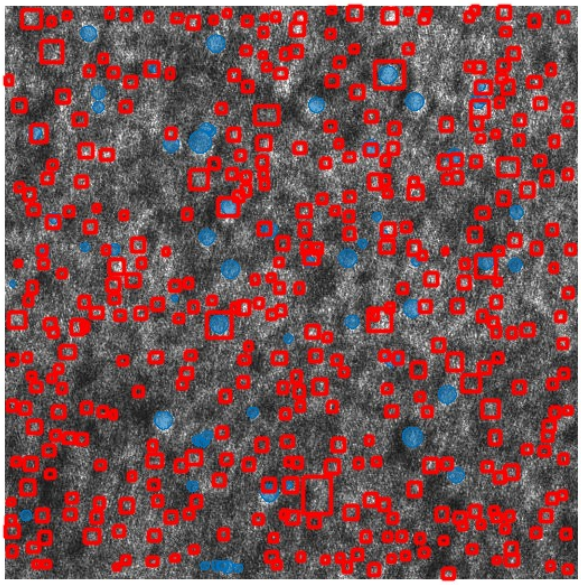


Figure 4.6: Similar to figure 4.5, the amount of non-existing small cavities is remarkable

Density and Size Estimation: Most images show some level of error in density and size estimation, but the model performs reasonably well in maintaining these errors within acceptable ranges for other high-performing images.

Bias in Predictions: The total true/pred/found columns help in understanding whether the model overestimates or underestimates the objects. For example, images 01, 10 and 16 have an over-prediction bias, while images 17 and 18 are under-predicted.

4.3.3 Custom Swelling Loss Function

In this section, a complete and comprehensive explanation of the creation of the custom loss function will be provided. The idea behind the creation of this extra component in the loss function stems from the belief that it will help the Mask R-CNN model to better estimate the swelling of images and to hopefully achieve better results.

Similarly as done before (4.3), Matlab installation files will be edited. In particular, during training, a fixed, hardcoded loss function is used. This loss function only takes into account mislabeled instances, mislabeled pixels.

Incorporating Swelling Indicators

Swelling indicators are key measurements that show how much alloys swell after being exposed to radiation. These measurements provide fundamental insights into how materials behave under radiation, ensuring the safety and durability of nuclear reactors. Incorporating swelling indicators into the loss function aims to improve the model's ability to accurately detect and measure these cavities. The MaskRCNNLoss.m loss function script is specifically modified to hopefully improve the conventional segmentation loss by incorporating indicators for swelling. The main components of this customized loss function are:

- **Segmentation Loss:** This is the standard loss component that measures the accuracy of the predicted masks compared to the ground truth masks. It includes binary cross-entropy and dice loss for pixel-wise classification. This part remains unchanged
- **Swelling Indicator Loss:** This new component specifically targets the accuracy of swelling predictions. Bearing in mind the formulas presented in 4.2, the estimated ground truth volume(s) and the predicted volume(s) are easily calculated since the loss function has total access to the ground truth data and to the prediction data. The swelling error (=loss) is again calculated using the aforementioned formulas.

The custom loss function can be mathematically expressed like this:

$$\text{Total Loss} = \alpha \cdot \text{Segmentation Loss} + \beta \cdot \text{Swelling Indicator Loss}$$

where α and β are weighting factors that balance the contributions of the segmentation loss and the swelling indicator loss. As far as α is concerned, it remained untouched because it is normalized based on the amount of received ground truth + predicted data. For β, the swelling loss by itself is not normalized with respect to all data, hence β is set to be the inverse of the number of predicted masks, in order to obtain coherence with the values already calculated by default. Empirically, it is considered to be a good scaling factor.

4.3.4 Results for Custom Loss Function

Due to the unavailability of required hardware to run the training, it was not possible to obtain the results for the custom loss function. However an attempt was still made, with a sort of toy example. This was the procedure:

- Instead of training a new model based on the real image sizes ([512 512 3]), a much smaller input was chosen, [128 128 3]. This was executed on the local computer and such input size made the network run without encountering GPU memory issues, and it allowed for a relatively fast training.
- Using this procedure, the custom loss function was debugged successfully. By constantly checking the value of the training loss, the normalization factor β was implemented. Still during training, it was indeed noticed a considerable difference in the training loss, indicating that the changes to the loss function actually took place.
- The results on the test set obtained by this model are not comparable given the fact that it's a reduced version of the original network.

One can make several predictions about the outcome of implementing the new custom loss function, regarding the metrics:

1. Precision, Recall and F1 score might change. For example, by implementing the swelling loss, the amount of false positives might decrease because necessarily, in order to obtain the same amount of swelling in output of the model, there will not be a great number of false positives because these would contribute to the swelling directly. However this might also mean that fewer instances are found, giving a lower Recall score.
2. The density error should also decrease, or maybe the amount of detected instances will be lower than the actual amount of cavities, indicating a negative density percentage error. This again makes sense if you assume a lower number of false positives.
3. The swelling error will naturally tend towards 0, because it is directly optimized at training time.

Image	precision	recall	overall F1	total true	total pred	total found
01.jpg	0,8605	0,6852	0,7629	54	43	37
02.jpg	0,8833	0,8413	0,8618	63	60	53
03.jpg	0,6444	0,8286	0,7250	35	45	29
04.jpg	0,7797	0,6970	0,7360	66	59	46
05.jpg	0,8261	0,8636	0,8444	66	69	57
06.jpg	0,8000	0,8727	0,8348	55	60	48
07.jpg	0,9167	0,8800	0,8980	25	24	22
08.jpg	0,9355	0,7632	0,8406	38	31	29
09.jpg	0,8846	0,7667	0,8214	60	52	46
10.jpg	0,3034	0,4500	0,3624	60	89	27
11.jpg	0,7188	0,7143	0,7165	161	160	115
12.jpg	0,6770	0,8596	0,7574	178	226	153
13.jpg	0,6200	0,7515	0,6795	165	200	124
14.jpg	0,7200	0,7347	0,7273	147	150	108
15.jpg	1,0000	1,0000	1,0000	26	26	26
16.jpg	0,2809	0,2852	0,2830	263	267	75
17.jpg	0,9811	0,6814	0,8043	609	423	415
18.jpg	0,9100	0,4561	0,6077	399	200	182
19.jpg	0,6769	0,5641	0,6154	312	260	176

Figure 4.7: Standard metrics of the baseline paper

4.4 Comparative Analysis with Base Paper

It is now possible to compare the results obtained between the Matlab implementation of Mask R-CNN and the Mask R-CNN python implementation of the baseline paper. The reported results they presented in [1] are visible in the pictures 4.7 and 4.8.

Quantitative metric comparison:

- **Precision and Recall:** to start things off, the paper reports precision and recall metrics the same way this thesis proposed earlier. Their overall precision spans from 0.28 up to 1.00, indicating some problems in detecting precise instances in few cases, however still robust. Similarly for recall, recall spans from 0.45 up to 1.00. The mean F1 score across all images is 0.73 (73%), while precision's and recall's respective mean values are 0.76 and 0.72.
- **Density and Size errors:** the overall density and size errors are relatively low, meaning that the model is able to provide a great estimation for both of these metrics, either by getting the density right or not being further away than a 50% from the true density.
- **Swelling Error:** the reported per-image percent swelling error returns great errors in absolute value, however often times the model actually un-

Image	density error	size error	swelling percent error
01.jpg	20,3704	5,4128	39,9193
02.jpg	4,7619	6,7400	20,8414
03.jpg	28,5714	2,9282	29,4261
04.jpg	10,6061	3,8126	20,1699
05.jpg	4,5455	4,7753	11,0614
06.jpg	9,0909	3,3670	3,2113
07.jpg	4,0000	2,2330	3,9952
08.jpg	18,4211	6,5627	3,2083
09.jpg	13,3333	4,3438	3,8937
10.jpg	48,3333	9,3361	19,3043
11.jpg	0,6211	3,3064	1,6813
12.jpg	26,9663	7,2113	0,9736
13.jpg	21,2121	9,9439	16,2639
14.jpg	2,0408	7,5868	30,9837
15.jpg	0,0000	2,6144	8,5678
16.jpg	1,5209	15,8234	32,9506
17.jpg	30,5419	3,0599	40,4378
18.jpg	49,8747	1,8328	48,8962
19.jpg	16,6667	5,7090	38,2073

Figure 4.8: Custom metrics of the baseline paper

derestimates the swelling of the material, leading to lower errors according to the formulas used. The reported swelling error ranges from -50% up to +20%.

Qualitative metric comparison:

- **Precision and Recall:** the results obtained in the Matlab implementation are slightly worse regarding precision and F1 score. This is due to the presence of plenty of false positives that drastically reduce the precision metric value, for a very slight increase in the recall metric. In the thesis version of Mask R-CNN, the mean average precision almost reached 0.50 (23% difference with the baseline precision), while the recall value is slightly off 0.81 (5% better than the baseline) and the mean F1 score is 14% worse, at roughly 0.59.

- **Density and Size errors:** in the Matlab model, the estimated density offered worse results. This is again due to the presence of a high amount of false positives which in turn increase the overall density of an image, since for the model it contains a lot of cavities when there are actually fewer. The size error cannot be compared to the baseline paper because in

the original paper they calculated the size errors cavity by cavity, and not by using the image area.

- **Swelling Error:** this error also got accentuated by the presence of false positives. In fact, while the baseline paper on average detects less instances than the actual amount of instances, the Matlab model detects too many of them, leading to an inevitable higher density estimation.

4.5 DeepLabV3+ Results

DeepLabV3+ is an advanced deep learning model designed for semantic image segmentation. The idea behind using DeepLabV3+ is to train this model using the same training images used for Mask R-CNN, in order to obtain a segmentation of the images. This segmentation does not contain additional information such as bounding boxes, instead it simply classifies each pixel to whatever classes the network is trained on.

In this instance, setting up the training was relatively simple, as already described briefly in 3.4. DeepLabV3+ is based on a ResNet, by leveraging ResNet as its backbone, DeepLabv3+ inherits the robustness and efficiency of residual learning, enabling it to perform accurate and detailed segmentation even in deep networks. This combination allows DeepLabv3+ to capture rich contextual information while maintaining high-resolution details, which is crucial for precise boundary delineation in segmentation tasks. The integration of ResNet into DeepLabv3+ enhances its ability to balance feature extraction and computational efficiency. Two important tests were run:

- ResNet-18 backbone. This is a great compromise for easy tasks, since it offers great precision and great training speed, as well as optimal inference time executions. The downside of using this architecture is the limited capability of learning complex patterns in hard datasets. In the CNL dataset case, ResNet-18 did not obtain amazing precision scores, therefore it was discarded and moved to the second test.
- In the second test, ResNet-50 was adopted as the backbone for DeepLabV3+. A bigger architecture implies longer training times and execution speed, for the upside of better precision and segmentation results in output. Usually, bigger architectures are more suited for offline tasks, contrary to real-time applications. Despite the choice of a bigger network, the model still doesn't seem to learn enough from the data, despite reaching the validation loss criterion, indicating a halt in the improvements on the validation set.

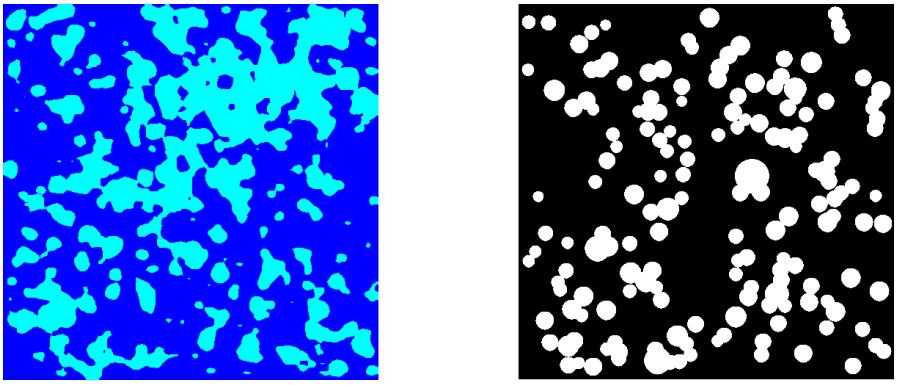


Figure 4.9: Output of DeepLabV3+ segmentation compared to Ground Truth binary mask, of the validation set

- To mention a few parameters, the standard input size was used, corresponding to the image sizes provided in the dataset. Data augmentation in the form of translation was applied, with a starting learning rate of 0.001, decaying every 6 epochs, for a maximum training time of 36 epochs.

The image 4.9 displays a sample segmentation of an image in the validation set, using the ResNet-50 version of the model. It is clear that the model struggles to pin down the exact region of a cavity, often joining cavities together, hence the obtained segmentations cannot be exploited in any form by Mask R-CNN.

In this case, it was also possible to draw a plot at training time, shown in figure 4.10. The orange line represents the validation loss, while the blue line represents the training loss. As one can see, the blue line contains a lot of spikes, probably indicating that the model struggles to smoothly converge towards a minimum, regardless of the training options. This behaviour was actually also observed during the training of Mask R-CNN, it is supposed that this uncommon behavior has to do with the complexity of the data at hand.

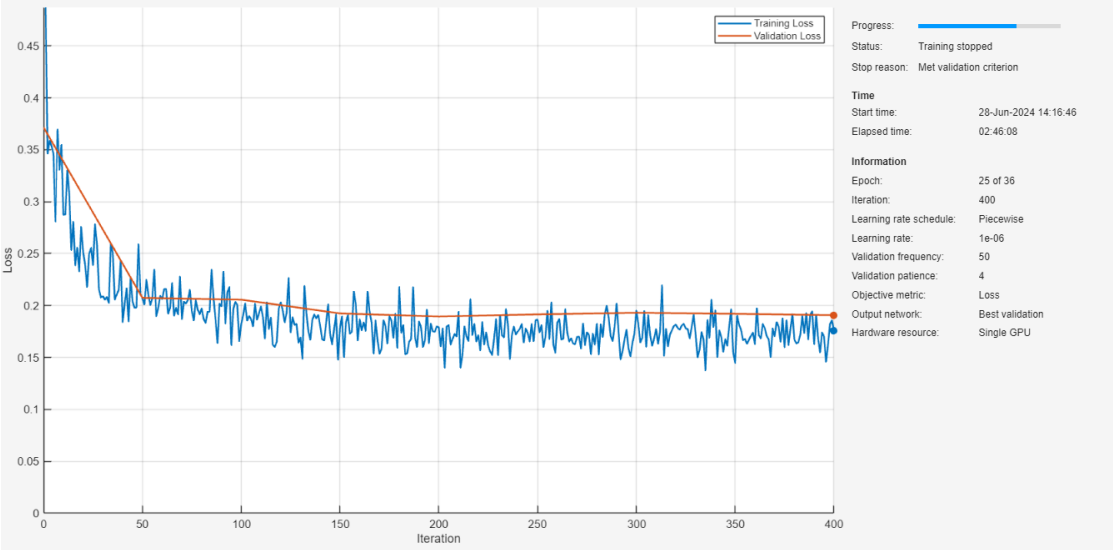


Figure 4.10: Matlab plot of the training, DeepLabV3+ model

4.6 SOLOv2 EXPERIMENTS

Few experiments were conducted using the SOLOv2 model, provided by Matlab. Having set the data up for Mask R-CNN already, all it took to set up SOLOv2 for training was simply editing a couple of lines of code. A short training was run on the local computer, since this model requires less GPU memory to be trained and it is definitely faster than Mask R-CNN while training. However, given the low amount of resources dedicated and given that SOLOv2 is outside the scope of this thesis, no long experiments were conducted. Moreover, the results obtained were not comparable in any shape or form with Mask R-CNN.

While it is undoubtable that SOLOv2 can achieve great results, it is definitely material for someone who wants to pick up this architecture and begin exploring the intricacies offered by the CNL and NOME datasets; leaving eventual progress, improvements and implementations to the reader.

Nevertheless, a training plot image is still provided in figure 4.11. A very spiky behavior is observed, again similar to how Mask R-CNN behaved at training time. Not much parameter tuning was done, also it is visible how the validation set loss begins worsening after a while, indicating overfitting.

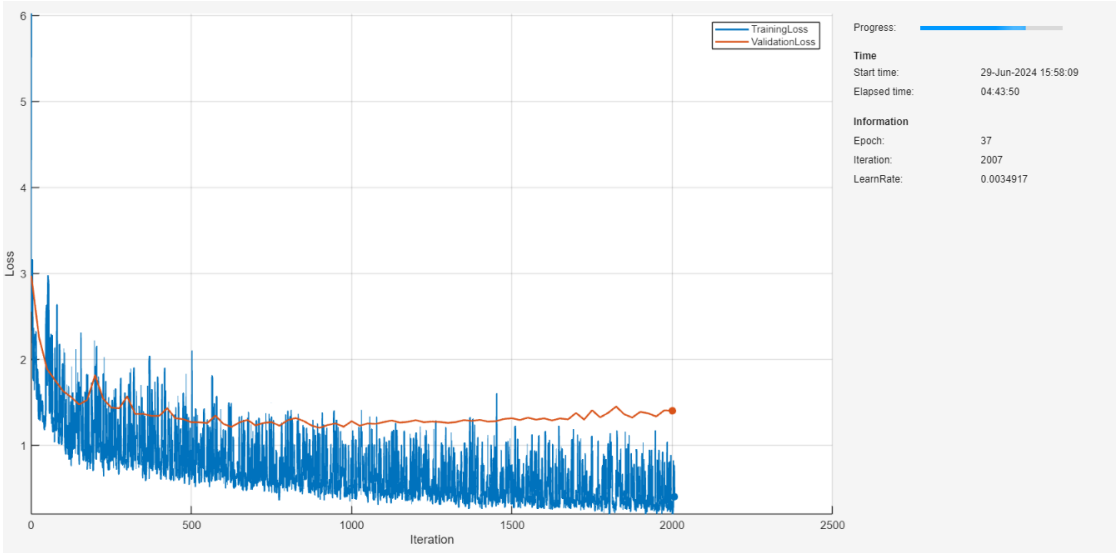


Figure 4.11: Matlab plot of the SOLOv2 training.

5

Conclusions and Future Works

This thesis demonstrates significant advancements in the automated measurement of material swelling in nuclear reactor alloys through the use of sophisticated deep learning models in Matlab. By employing Mask R-CNN and creating custom loss functions tailored to detect swelling indicators, a highly reliable and precise method for analyzing electron microscopy images has been developed. Thus it is possible to understand irradiated material behaviors, by adopting deep learning techniques in materials science.

The experimental results indicate that the Mask R-CNN model acquired a somewhat good F1 score, precision, and recall rates, demonstrating its utility in detecting and segmenting cavities. Nevertheless, it was seen that the model still generated a substantial amount of inaccuracies. Let's recap and breakdown potential improvements for future research.

1. False Positives. At inference time it was noticed that on several images from the test set many false positives were found. Despite the use of non maximum suppression, in some cases false positives were still abundant. Especially when compared with the baseline paper and results, it is possible to improve. Going back to the tables (figure 4.3) containing the segmentation results, images such as number 15 can be improved a lot. Perhaps this needs to be fixed at training time, in some ways.

2. Metric tuning. Some metrics might not be final and might need some tuning. For example, the swelling loss and the size error metrics can benefit with further tuning on how they are calculated. For the sizes proposed, maybe it is more suited to take into account errors cavity by cavity instead of using the

total area.

3. Custom Loss Function Implementation. The loss function has not been refined extensively. It might need more tuning with the parameters that balance the weight of each loss component, plus the way the swelling loss is calculated might need some extra refinement. Future research can be done in this area for this specific task.

4. Segmentation Models for the background. The results of DeepLabV3+ were not satisfactory. However there might still be potential in this technique. For example, in Matlab there exists a possibility to use ResNet-101, an even bigger architecture as a backbone for DeepLabV3+, instead of ResNet-50. Future research should look into this problem, and then use the ground truth cavity segmentation provided by DeepLabV3+ ran on the training images.

5. Reworking the dataset. Not a lot of attempts were spent in looking for subsets of the training set to seek for overall improvements. It is believed that using less data but more efficiently from the training set could improve the model generalization capabilities.

6. Two-Round Fine-Tuning with Loosely Similar Datasets. The advantages of using this technique is to optimize the Mask R-CNN model by using similar datasets, for example the dataset provided by NOME. Using weakly supervised ways to manage the training process might improve the performance of the network thanks to broader datasets. It hasn't been tested but there is a possibility it might improve the accuracy and robustness of the model.

7. SOLOv2: SOLOv2 also has some potential regarding instance segmentations. It is an inherently different architecture that performs the same job as Mask R-CNN. However this model hasn't been looked into in this thesis, and it offers its own collection of tunable parameters before training.

8. Different applications. It is also interesting to leverage these models to learn on different datasets, it is recommendable to explore new datasets to learn more about the capabilities of these models.

9. Wait for updates. Perhaps in the future bigger Mask R-CNN models will be adopted in Matlab, therefore making some difficult tasks, such as this one, easier.

References

[1] Ryan Jacobs et al. "Materials swelling revealed through automated semantic segmentation of cavities in electron microscopy images". In: *Scientific Reports* 13 (Mar. 2023).

[2] Nasr M. Ghoniem and Yinan Cui. "1.22 - Dislocation Dynamics Simulations of Defects in Irradiated Materials". In: ed. by Rudy J.M. Konings and Roger E. Stoller. Second Edition. Oxford: Elsevier, 2020, pp. 689–716. ISBN: 978-0-08-102866-7.

[3] Chris M. Anderson et al. "Automated Detection of Helium Bubbles in Irradiated X-750". In: *Ultramicroscopy* 217 (2020), p. 113068.

[4] Wei-Ying Chen et al. "In-situ TEM investigation of void swelling in nickel under irradiation with analysis aided by computer vision". In: *Acta Materialia* 254 (2023), p. 119013.

[5] A.A. Stepashkin and N.Yu. Nikitin. "Statistical analysis, regression, and neural network modeling of the tensile strength of thermoplastic unidirectional carbon fiber-polysulfone composites". In: *Carbon Trends* 15 (2024), p. 100368.

[6] Islam M. Mantawy and Naga Lakshmi Chittitalli Ravuri. "Predicting low-cycle fatigue-induced fracture in reinforcing bars: A CNN-based approach". In: *Structures* 64 (2024), p. 106509.

[7] Graham Roberts et al. "Deep Learning for Semantic Segmentation of Defects in Advanced STEM Images of Steels". In: *Scientific Reports* 9 (Sept. 2019).

[8] Stephen Taller, Luke Scime, and Ty Austin. "A new paradigm in electron microscopy: Automated microstructure analysis utilizing a dynamic segmentation convolutional neutral network". In: *Materials Today Advances* 21 (2024), p. 100468.

[9] Junlin Lin et al. “Transregional spatial correlation revealed by deep learning and implications for material characterisation and reconstruction”. In: *Materials Characterization* 178 (2021), p. 111268.

[10] Rajat Sainju et al. “DefectTrack: a deep learning-based multi-object tracking algorithm for quantitative defect analysis of in-situ TEM videos in real-time”. In: *Scientific Reports* 12 (Sept. 2022).

[11] Surya Prakash Mishra and M.R. Rahul. “A comparative study and development of a novel deep learning architecture for accelerated identification of microstructure in materials science”. In: *Computational Materials Science* 200 (2021), p. 110815.

[12] Kaiming He et al. “Mask R-CNN”. In: *CoRR* abs/1703.06870 (2017).

[13] Lun Che et al. “Deep learning in alloy material microstructures: Application and prospects”. In: *Materials Today Communications* 37 (2023), p. 107531.

[14] Xinlong Wang et al. *SOLOv2: Dynamic and Fast Instance Segmentation*. 2020.

[15] Wei Wei et al. “Augmenting progress monitoring in soil-foundation construction utilizing SOLOv2-based instance segmentation and visual BIM representation”. In: *Automation in Construction* 155 (2023), p. 105048.

[16] Ryan Cohn et al. *Instance Segmentation for Direct Measurements of Satellites in Metal Powders and Automated Microstructural Characterization from Image Data*. 2021.

[17] Qing Yang et al. “Road Scene Instance Segmentation Based on Improved SOLOv2”. In: *Electronics* 12.19 (2023).

[18] L. Minh Dang et al. “Lightweight pixel-level semantic segmentation and analysis for sewer defects using deep learning”. In: *Construction and Building Materials* 371 (2023), p. 130792.

[19] Kavindu Wijesinghe et al. “Characterization of microscopic deformation of materials using deep learning algorithms”. In: *Materials Design* 208 (2021), p. 109926.

[20] Leon Derczynski. “Complementarity, F-score, and NLP Evaluation”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by Nicoletta Calzolari et al. May 2016.

Acknowledgments