ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MASTER'S DEGREE IN ARTIFICIAL INTELLIGENCE

Mitigating the effects of Severe Imbalance in Multi-class Semantic Segmentation

MASTER THESIS IN

MACHINE LEARNING FOR COMPUTER VISION

CANDIDATE

Giuseppe Morgese

ACADEMIC SUPERVISOR:

Prof. Samuele Salti

RESEARCH SUPERVISORS:

Daniel Soukup

Lukas Bednar

MARCH 2024

ACADEMIC YEAR 2022/2023

Abstract

Class imbalance is one of the main weaknesses in modern machine learning methods. In this area, datasets with an imbalance ratio greater than 1:100 are defined as severely imbalanced. These require specific precautions and techniques to deal with the issue.

In this thesis, different approaches to tackle the problem of severely imbalanced datasets in semantic segmentation are explored. Solutions such as resampling, the One-vs-Rest approach, and loss change are implemented and compared discussing their benefits and drawbacks. Furthermore, the delicate evaluation process is explained in all its complexity giving specific weight to the obtained results.

Dedication

To all students who have been victims of suicide.

A tutti gli studenti vittime di suicidio.

Contents

List of Figures

List of Tables

Acronyms

ADASYN Adaptive Synthetic Sampling. 6

CE Cross Entropy. 26, 29

CNN Convolutional Neural Network. 4, 24

FCN Fully Convolutional Network. 4

FL Focal Loss. 26

FPR False Positive Rate. 5, 32

GIMP GNU Image Manipulation Program. 11

ICF Inverse Class Frequency. 26, 50

ISCF Inverse Square root Class Frequency. 26, 30, 50

OHE One-Hot-Encoded. 20, 22

PSoI Performance Stability over (later) Iterations. 33, 38

PSoMT Performance Stability over Multiple Training. 33, 38

ROI Region of Interest. 15, 22, 27

SMOTE Synthetic Minority Over-sampling Technique. 6

TPR True Positive Rate. 5, 32

wCE Weighted Cross Entropy. 26, 29

Chapter 1

Introduction

The advent of machine learning has revolutionized numerous fields, from image recognition to natural language processing, and has the potential to deliver solutions to many complex problems. However, one of the significant challenges that persist in the field of machine learning is the issue of class imbalance. This problem is particularly pronounced in datasets where the imbalance ratio exceeds 1:100, categorizing them as severely imbalanced. The presence of such severe class imbalance can significantly skew the performance of machine learning models, necessitating the need for specific precautions and techniques to effectively handle this issue.

This thesis delves into the problem of severe class imbalance in the context of semantic segmentation. Semantic segmentation, a crucial task in many computer vision applications, involves classifying each pixel in an image to a particular class. However, the presence of severe class imbalance can drastically affect the performance of semantic segmentation models, leading to biased predictions toward the majority class.

Various approaches to tackle this problem are explored in this thesis. A dataset of industrial high-resolution images of battery foils with manual annotations of four types of defects is utilized. Solutions such as resampling, the One-vs-Rest approach, and loss change are implemented and compared, each with unique benefits and drawbacks. Resampling techniques attempt to balance the class distribution by either oversampling the minority classes or undersampling the majority class.

The One-vs-Rest approach involves modifying the problem formulation to alleviate the effects of class imbalance, while loss change refers to the modification of the loss function to give more weight to the minority classes.

The thesis also investigates the intricate process of evaluating the performance of models trained on severely imbalanced datasets in semantic segmentation. The evaluation process is complex and requires carefulness. The models are evaluated using various quantitative metrics and compared with qualitative analysis. This process is discussed in detail, giving specific weight to the results obtained from the implemented solutions. The results have indicated that some of the explored methods can enhance the segmentation accuracy and robustness, particularly for the minority classes. The limitations and challenges of the methods are also discussed, and some directions for future work are suggested.

This work aims to provide a comprehensive understanding of the problem of severe class imbalance in semantic segmentation and offers effective strategies to mitigate its effects. It is hoped that this thesis will serve as a valuable resource for researchers and practitioners in the field, aiding them in their quest to develop robust and fair machine learning models.

The thesis is organized as follows: Chapter 2 provides a background on semantic segmentation, convolutional neural networks, imbalanced datasets, and evaluation metrics. It also reviews some of the existing methods and challenges in the field. Chapter 3 describes the dataset, the model architecture, and the various approaches explored in this study. Chapter 4 presents and analyzes the results obtained from the evaluation of the models. Chapter 5 discusses the implications, limitations, and challenges of the methods and the results. Chapter 6 summarizes the main findings and contributions of the study and provides some final remarks.

Chapter 2

Literature Review

This section provides an overview of relevant topics necessary for understanding subsequent sections, including the semantic segmentation task, implemented architecture, imbalanced datasets in machine learning, and evaluation metrics for imbalanced classification.

2.1 Semantic Segmentation

Semantic image segmentation is a computer vision task that involves labeling each pixel in an image with a predefined set of classes. This technique is vital for image analysis as it facilitates the description, categorization, and visualization of regions of interest in an image. This task is useful for various applications, such as autonomous driving, medical imaging, scene understanding, and more [6].

Semantic image segmentation faces several common challenges. One of these is the need for processing images at a high resolution to preserve fine-grained details and boundaries, which also increases the computational cost and memory requirements of the models. Another challenge is the potential class imbalance in an image, where some semantic classes may be more frequent or dominant than others, such as the background or the sky. This can cause the models to be biased towards the majority classes and ignore the minority ones, leading to poor performance on rare or small objects. Additionally, semantic image segmentation usually has to deal with a wide range of variability and diversity in the images, such as different lighting conditions, occlusions, viewpoints, scales, shapes, textures, and more. This makes the task more challenging and requires the models to be robust and generalizable to different scenarios.

These are just a few of the many challenges faced in semantic image segmentation, and ongoing research continues to address these and other issues in the field. For instance, in the context of imbalanced datasets, domain adaptation and the use of unsupervised learning techniques could be particularly useful as they can learn robust features that are not affected by the less frequent classes. However, further research is needed to improve the performance of these models on imbalanced datasets [6].

2.2 CNNs in Semantic Segmentation

Convolutional Neural Networks (CNNs) are a type of Deep Learning neural network architecture commonly used in Computer Vision. CNNs are primarily used for tasks like image classification, object detection, and image generation. They consist of layers of artificial neurons called nodes, which calculate the weighted sum of the inputs and return an activation map [12]. In the context of semantic segmentation, CNNs typically use a Fully Convolutional Network (FCN) architecture, which replaces the fully connected layers of a traditional CNN with convolutional layers. This allows the network to process input images of any size and produce a corresponding output map of the same size, where each pixel is assigned a label [15].

U-Net is a convolutional neural network developed for biomedical image segmentation [17]. The network is based on an FCN whose architecture is modified and extended to work with fewer training images and yield more precise segmentation. The U-Net architecture consists of a contracting path and an expansive path, making it particularly effective for tasks requiring precise localization.

U-Net has several advantages over other segmentation models. It is flexible and can be used for any image masking task. It has high accuracy given proper training dataset and training time. It does not contain any fully connected layers, which makes it more accurate. Moreover, U-Net is still particularly widespread in the context of semantic segmentation, and its design still represents a reliable starting point for new architectures [21] [9] [18].

2.3 Imbalanced Datasets in Machine Learning

In machine learning, we often encounter imbalanced datasets where one class has considerably fewer instances than the other. Many machine learning algorithms assume an equilibrium between majority and minority classes, leading to a sub-optimal performance on imbalanced data. This imbalance can skew the outcomes for each metric, so testing a model's performance across many metrics is key for determining how well a model works. In the context of semantic image segmentation tasks, the labels assigned to each pixel in an image might also be imbalanced. This could occur if certain objects or features appear less frequently in the images. This imbalance can pose challenges for models like U-Net, which might struggle to accurately segment the minority classes.

Evaluation metrics for imbalanced classification problems [3] are a common challenge. In this regard, the case of Accuracy conveys the idea very well. While commonly used in classification tasks, this metric can be misleading in imbalanced datasets as it may overemphasize the performance of the majority class while neglecting the minority class. Thus, other metrics like F-scores, True Positive Rate (TPR), and Precision, provide a more comprehensive assessment of model performance across different classes and help mitigate the impact of class imbalance on evaluation. In particular, F1 score, F0.5 score, F2 score, and TPR are commonly used for imbalanced classification. These metrics are particularly relevant when evaluating the performance of semantic segmentation models on imbalanced datasets. Even though all the listed metrics can be used in similar contexts, the choice between using TPR/False Positive Rate (FPR) trade-off and F-scores, for example, depends on factors such as the cost of false positives and false negatives and the imbalance in the dataset. In fact, F-scores are based on the harmonic mean of precision and recall, which assumes that false positives and false negatives have the same impact. However, in some cases, one type of error may be more costly or undesirable than the other. In such cases, using a TPR/FPR trade-off can help to balance the sensitivity and specificity of the models and adjust the decision threshold according to the desired trade-off.

F-scores can be misleading also when the dataset is imbalanced. For instance, in a dataset where 99% of the samples are negative and 1% are positive, a model that always predicts negative will have a high F-score, even though it is useless for the positive class. In such cases, using the TPR/FPR trade-off can help to evaluate the performance of the models on both classes and avoid bias towards the majority class.

Understanding the limitations of metrics like F-scores in imbalanced datasets points out the importance of exploring alternative evaluation approaches. The TPR/FPR trade-off offers a slightly different perspective, particularly when considering the trade-offs between sensitivity and specificity in classification tasks.

2.4 Data resampling

As shown by the literature, resampling can be a valid option to overcome the imbalance problem in different tasks. There are mainly 3 options: random undersampling, random oversampling, and synthetic data generation [3]. Random undersampling reduces the number of majority class instances, while random oversampling replicates the minority class instances. Synthetic data generation creates new instances of the minority class using techniques such as SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic Sampling). SMOTE [5] creates synthetic samples by selecting a minority class sample and one of its nearest neighbors. A new sample is then generated by interpolating between these two points. This process is repeated until the desired number of synthetic samples is created. ADASYN [8], instead, generates synthetic samples by selecting a minority class instance and its k nearest neighbors. It then calculates the class imbalance degree for each minority class sample. Based on this degree, synthetic examples are created by interpolating between the selected instance and its neighbors. The main difference between the two is that SMOTE generates samples uniformly across the feature space, while ADASYN adapts the number of synthetic samples based on the difficulty of classification for each instance.

Each option has its advantages and disadvantages, depending on the characteristics of the dataset and the learning algorithm. Synthetic data generation is not used in this work, as it has been preferred to focus more on basic solutions, but it represents an interesting possibility to tackle the problem of imbalance datasets.

Chapter 3

Methods

Serving as the cornerstone of this study, this section introduces and explains all methods used for the semantic segmentation of industrial images of battery foils. It covers the data, model, and techniques used in the study. First, the 3believe Dataset [10], which consists of high-resolution images of battery foils with manual annotations of four types of defects, is introduced and its challenges, as the severe class imbalance, and characteristics are discussed. Next, the model that serves as the starting point of this study, a U-Net architecture with some minor modifications, is also described along with its training procedure. Then, various resampling strategies to mitigate the class imbalance in the dataset, are explored and their advantages and disadvantages are discussed. After that, different variations of the baseline segmentation model, such as using a One-vs-Rest approach or changing the loss function, are investigated and their results are compared with the baseline model. Finally, the evaluation process is described, showing all the encountered challenges.

3.1 3believe Dataset

The 3believe dataset, used in this study, is designed for the inspection of car battery foils [10]. The work has received funds from the European Union's Horizon 2020 program, and it consists of real industrial images along with manual annotations. This dataset comprises 26 high-resolution images and their corresponding anno-

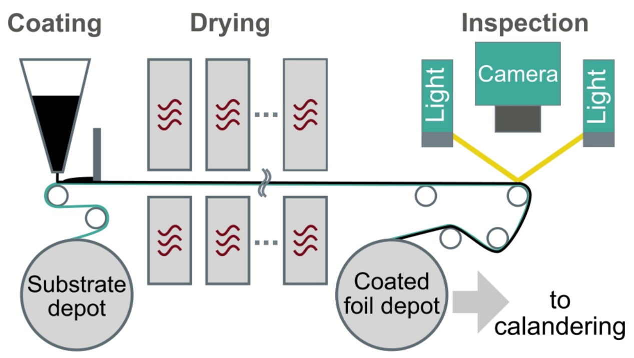


Figure 3.1: Inline electrode coating inspection process scheme. Taken directly from [10].

tations. An important factor influencing the electrical characteristics and safety of battery cells is the quality of the applied electrode material. This material is produced in the so-called coating process, in which a *slurry*, a mixture of active material, binder material, conductive additives, and solvents, is prepared and applied onto a metal substrate foil. Subsequently, a blade mounted over the substrate lets the slurry pass up to a defined thickness. Ideally, the resulting coating material is finely grained and fully covers the substrate area evenly. However, coating surfaces can deviate from this ideal conditions. Among others, a typical type of defect occurs when the blade gets clogged with agglomerates within the slurry mix, which leads to missing or unevenly applied coating behind the blade. Optical quality assurance can help in ensuring that only cathodes of high quality are used for battery cells. Figure 3.1 shows the acquisition system's position in the conveyor belt.

The images have been captured using the setup detailed in [10], which comprises four essential components: **(1)** a controller synchronizing camera acquisition, material motion, and control of lights, **(2)** a high-speed industrial camera, **(3)** four line light sources, and **(4)** a PC coordinating data acquisition and computing the foil surface representation.

The camera is positioned directly above the conveyor belt with a spatial resolution of 50 µm/px. The four light sources are strategically placed in a 4-orthogonal-configuration around the camera's field of view, with a 45° rotation angle relative to the transport direction and a polar angle of 55°. Each object point is captured

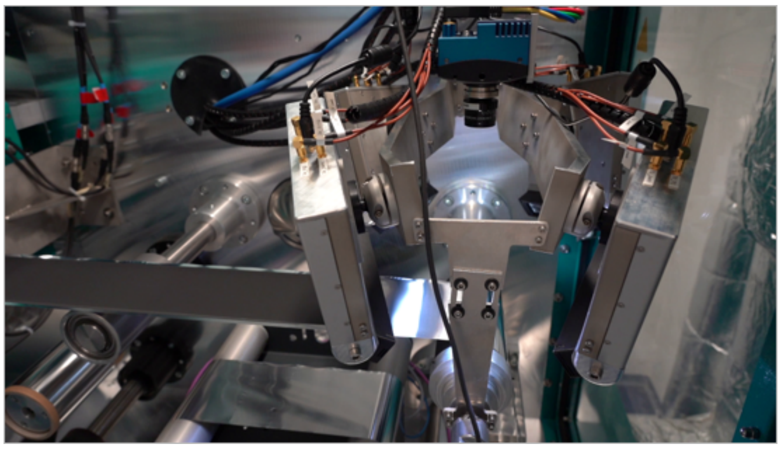


Figure 3.2: Acquisition System prototype.

four times under four different illumination directions. The prototype of the system is shown in Figure 3.2.

Subsequently, the acquired images undergo photometric stereo [22] processing to reconstruct the surface, as detailed in [11]. Photometric stereo relies on the Lambertian surface assumption, which posits that, under uniform illumination, the reflected intensity is independent of the viewing direction. In other words, a Lambertian surface would ideally appear equally bright when viewed from any direction and illuminated from any direction. Note that real-world surfaces may not adhere to this property. The actual intensity observed depends on the relative alignment of the light sources with respect to the surface. To estimate surface normals and albedo, the photometric stereo algorithm described in [1] has been used. Here is a brief, simplified overview of the method's initial assumption.

Lambertian reflectance under known light directions is considered: a surface is acquired from $n \geq 3$ known illumination directions from a fixed view point. Each acquisition has p x q pixels with $1 \leq i \leq p$ and $1 \leq j \leq q$ as row and column indices respectively. Given these premises:

$$\rho_{i,j}\mathbf{n}_{i,j} = \mathbf{L}^{+}\mathbf{o}_{i,j} \tag{3.1}$$

where $\rho_{i,j} \in \mathbb{R}$ is the albedo, $\mathbf{n}_{i,j} \in \mathbb{R}^3$ are the surface normals, $\mathbf{L}^{+} \in \mathbb{R}^{3\times n}$ is the Moore-Penrose pseudo inverse of the known illumination directions $\mathbf{L} \in$

$\mathbb{R}^{n\times 3} = (\mathbf{l}_1, ..., \mathbf{l}_n)^T$ and $\mathbf{o}_{i,j} \in \mathbb{R}^n$ are the observed intensities.

Once the surface normals have been estimated, it becomes possible to obtain the images that make up the dataset. In this framework, the primary goal is to precisely detect and distinguish between different types of defects. To this extent, semantic segmentation is considered the most suitable task and four different kinds of defects are defined:

- Agglomeration: Excess coating material
- Cavity: Small-diameter pores
- Crack: Cracks in the coating area
- Blade trail: Traces left by the blade on the coating area

Any remaining pixels are categorized as background. A more detailed explanation of these defects and their underlying causes can be found in [11], where they are referred to by different names. Figure 3.3 provides visual examples of each defect.

The indexed masks have been manually labelled by experts in our team using GNU Image Manipulation Program (GIMP) [20]. Some examples of surface normals image and respective annotations are shown in Figure 3.4.

The train/validation split is achieved by vertically cropping each image based on a specified percentage and starting region. For all experiments, a validation vertical split of 0.33 and the ”top” region have been selected. This means that, for each image, the first third has been reserved for validation purposes, while the remaining has been allocated for training.

The main issue of the dataset is the different severe class imbalances. Specifically, there is an imbalance between the positive and negative classes (i.e. defect and non-defect) as well as an inter-class imbalance between the defects. Therefore, devising a resampling procedure is not straightforward, it demands a more cautious approach. For a detailed breakdown of the class imbalance within the training set pixel-wise, please refer to Table 3.1. Meanwhile, Table 3.2 provides an overview of the pixel ratios between the different classes.

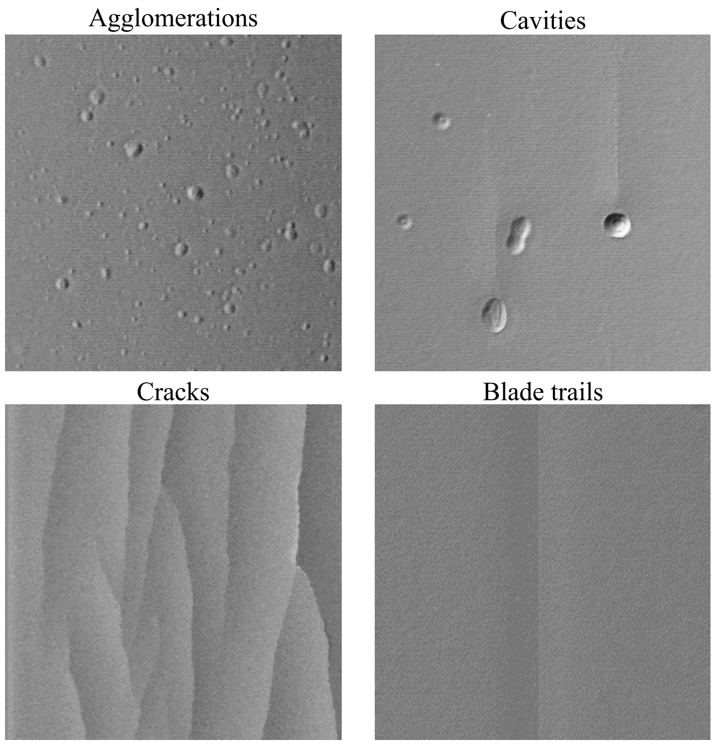


Figure 3.3: Defect examples in patches depicting the gradient in x-direction (derived via photometric stereo).

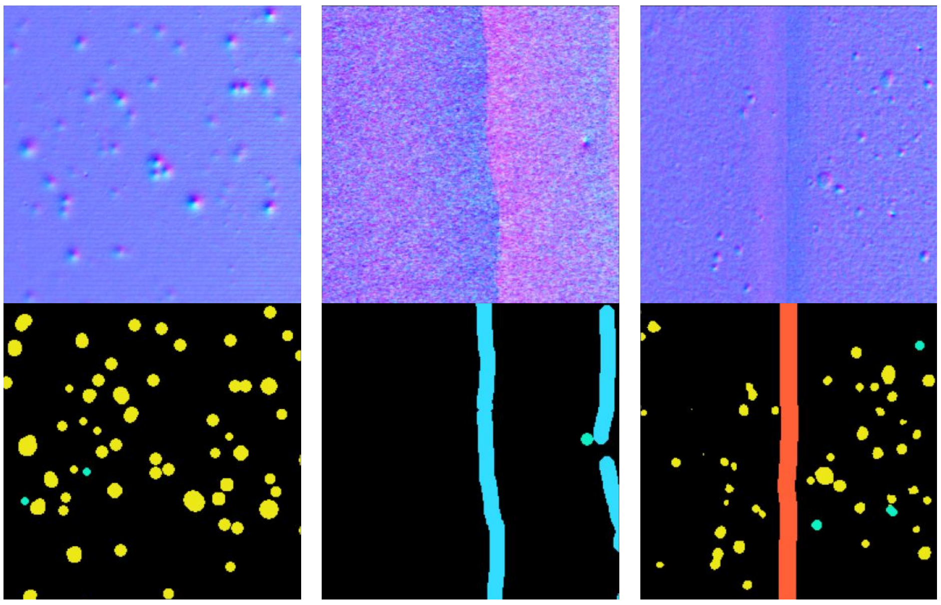


Figure 3.4: Examples of image/annotation pairs. In surface normals images **(top)**: red, green, and blue channels correspond to the x, y, and z coordinates of the surface normal. Red areas indicate normals pointing to the right, while blue areas represent normals pointing down. The color/defect association in annotations **(bottom)** is as follows: ⬛ background, 🟨 agglomeration, 🟩 cavity, 🟦 crack, 🟧 blade trail.

Negative Class	Positive Class			
Background	Agglomeration	Cavity	Crack	Blade trail
61,341,642	1,532,701	39,040	550,683	752,734
	2,875,158			

Table 3.1: Number of pixels in training set per class

	Cavity	Crack	Blade trail	Background
Agglomeration	39:1	3:1	2:1	1:40
Cavity		1:14	1:19	1:1571
Crack			3:4	1:111
Blade trail				1:82
Total defects				1:21

Table 3.2: Pixel ratios in training set

As can be seen in the tables, the cavity class exhibits the most pronounced pixel scarcity, making it the most challenging class to work with. Notably, the imbalance in relation to other positive classes is already evident, even if still relatively moderate. However, the imbalance between cavity and the background is notably high, with only one cavity pixel for every 1571 background pixels. Conversely, the segmentation model is expected to perform comparatively better on the agglomeration class, with respect to other positive classes, due to its substantial pixel count. Note that these observations are based solely on the dataset's composition and may vary depending on the implementation of imbalance mitigation strategies. Once the dataset nature has been discussed, the next natural step is to describe the other base component: the starting architecture.

3.2 Baseline Segmentation Model

The study has a model as a starting point from which different possible variations have been tried out. The starting model will from now on be referred to as the 'baseline' model. This baseline model is a U-Net, as described in the original paper [17], with a few minor modifications. These include using convolutional layers with one-pixel padding instead of zero-padding, thus employing a 3x3 convolution as the final layer (rather than a 1x1 convolution). Additionally, the model's first layer features 32 feature maps instead of the original 64 but the same feature channels scaling factors. It's important to note that no measures to address class imbalances have been incorporated into this model. The Cross Entropy Loss function has been initially employed as the model's loss function. Like every other model in this study, it has been trained for 10,000 iterations. No checkpoint selection strategies have been implemented since, as later shown in section 4.1, the models' performance remain stable after the first 7,000 iterations. Figure 3.5 shows the architecture of the baseline model.

Decisions regarding normalization and the optimizer, along with the subsequent hyperparameters, have been intentionally left unaltered being out of the scope of this thesis. Nevertheless, considering their relevance, further details are provided as follows. Batch normalization has been employed in the encoder, bridge, and decoder sections of the U-Net after each convolution, utilizing the following hyperparameters: *"momentum"*: 0.1 and ϵ: 0.001. In terms of the optimizer, AdamW [16] has been used with the following hyperparameters: *"learning rate"*: 0.001 and *"weight decay"*: 1e-05. To be fed to the network, each image-label pair undergoes various operations. The whole process is depicted in Figure 3.6. First of all, both the image and the label are cropped to isolate a Region of Interest (ROI), whose coordinates are specified by the image's annotator, defined to select areas of the image containing defects. Then, the label is converted from an RGB image to an indexed mask, where each class in the image is associated with a specific index. Specifically, the mapping is as follows: 0 to background, 1 to agglomeration, 2 to crack, 3 to cavity, and 4 to blade trail. Following this, the data is divided into training and validation regions, as already explained in section 3.1. After the train/validation split, 256 x 256 patches of both the image

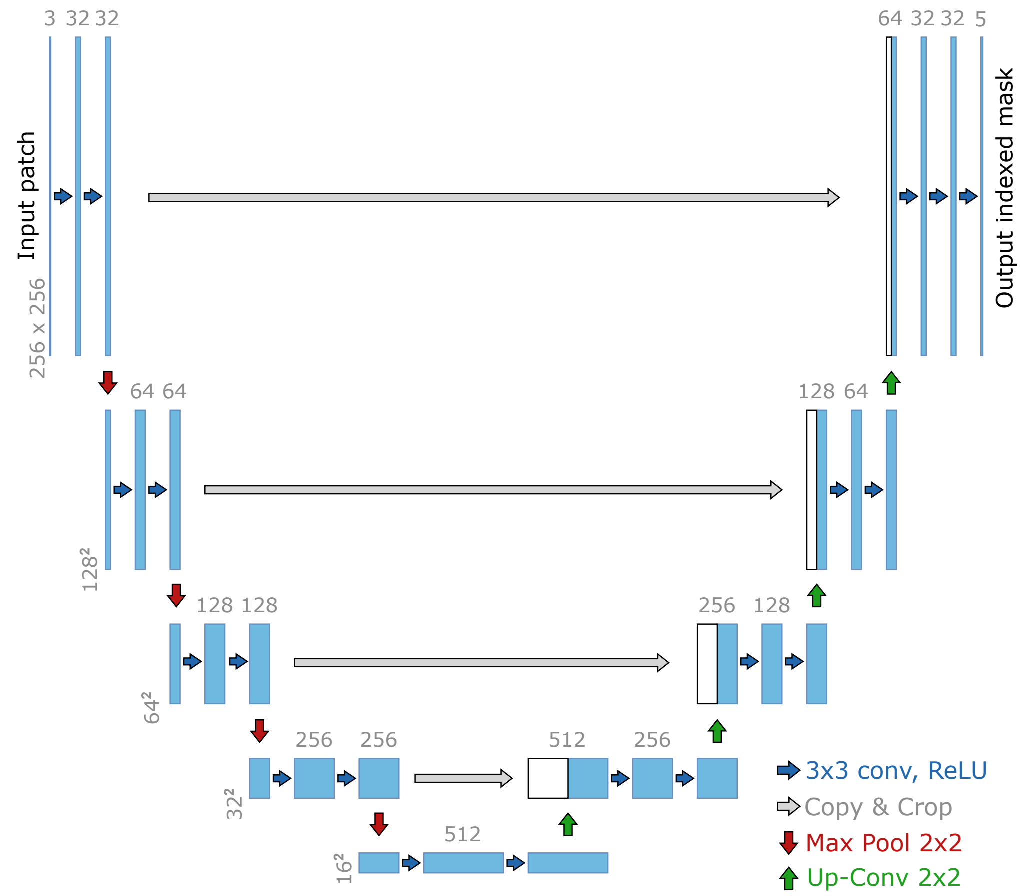


Figure 3.5: Network architecture. The schema has been adapted from the original U-Net shown in [17].

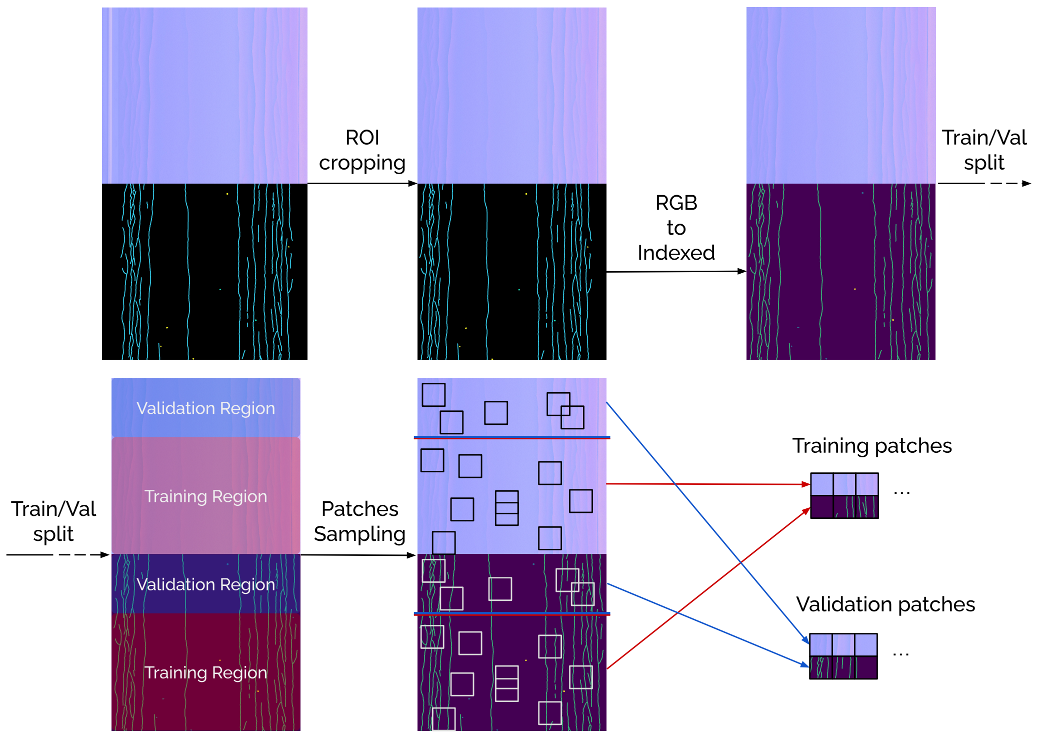


Figure 3.6: The sampling pipeline providing image/label patch pair samples used to train and infer the U-Net.

and the label are randomly sampled together from either the train or the validation region, depending on the phase. Note that the patches size has been chosen considering that the input images resolution usually exceeds 1200 x 1200. Both the training and validation patches are refreshed after a set number of iterations. This approach ensures that the model is not trained or validated on the same patch more than once. The total number of sampled training (or validation) patches is calculated as follows:

$$N_{patches} = N_{batch}\, iter_{ref} \tag{3.2}$$

where N_{batch} is the batch size and $iter_{ref}$ is the iteration at which the patches are refreshed.

Although some promising predictions have been obtained through the baseline model, as shown in Figure 3.7, the results have not been satisfactory on the validation set. Many defects are only partially detected or, in some cases, completely overlooked, as depicted in Figure 3.8. Note that no data augmentation techniques

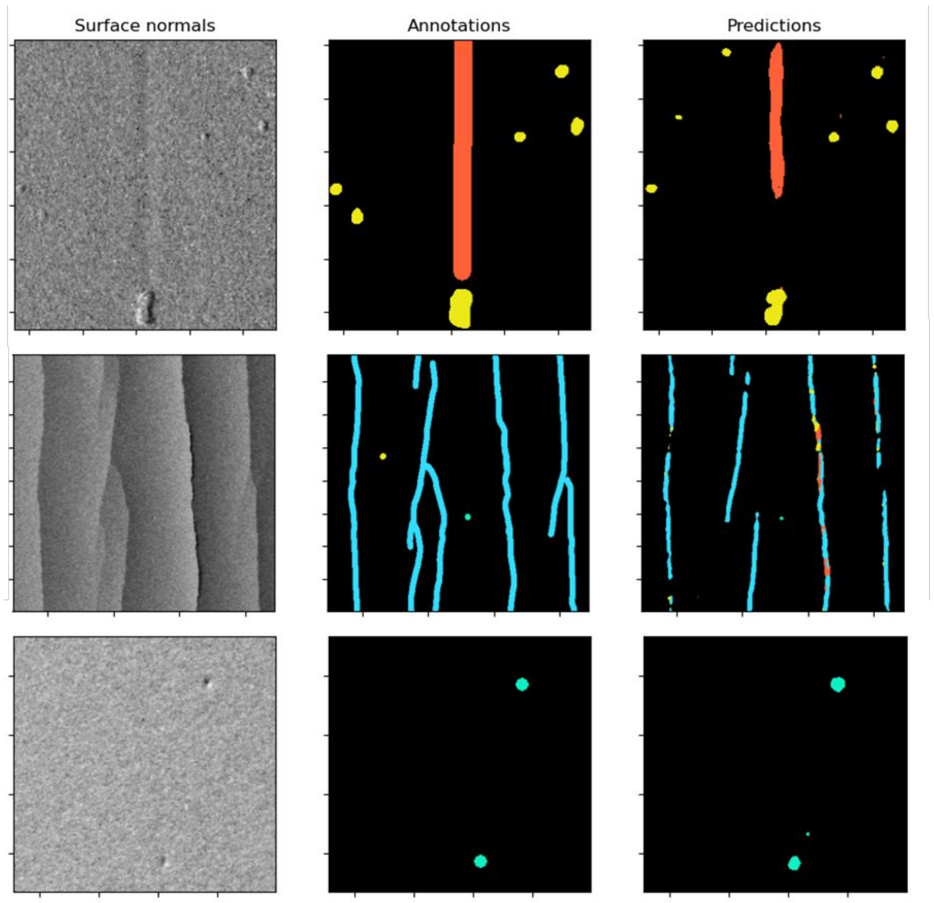


Figure 3.7: Examples of good results from baseline U-Net model.

have been used since they would require specific carefulness in an industrial setting like this one. Also, initial results on the majority classes have been satisfactory, thus shifting the efforts towards the problem of imbalance rather than on dataset's variance. Nonetheless, data augmentation could be an option worth exploring in future works. The upcoming sections describe selected possibilities explored for improvement.

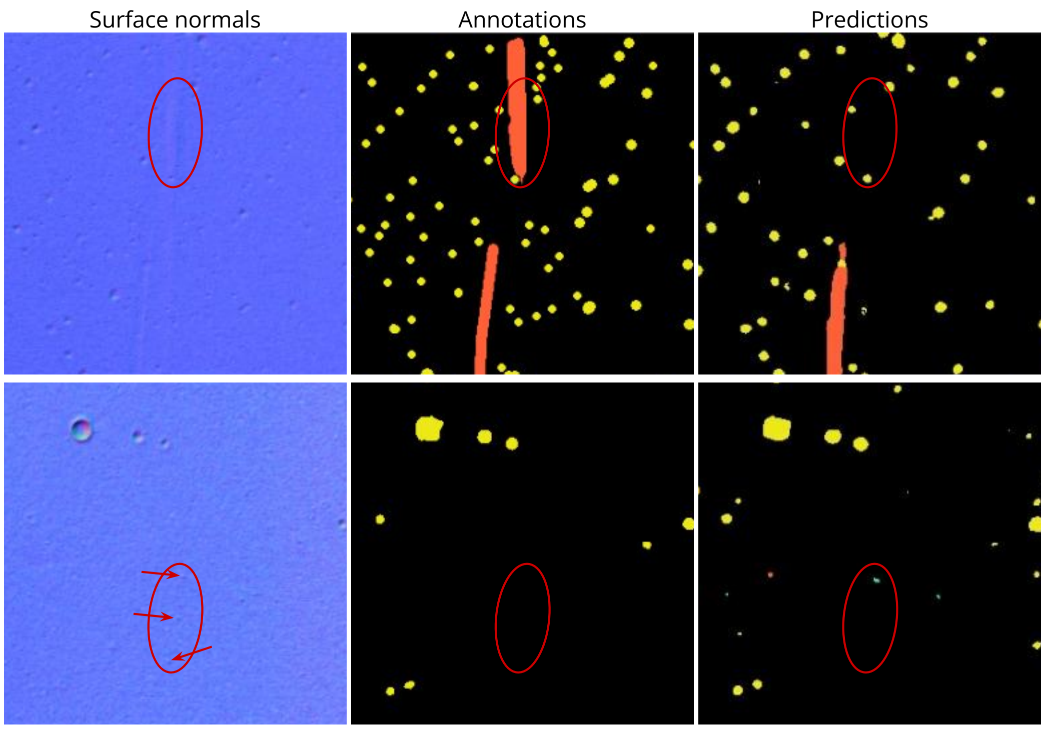


Figure 3.8: Baseline fails example. In the first prediction, a blade trail (orange) is completely overlooked (False Negative) while in the second case, only one cavity (aqua) out of three is detected (False Negatives).

3.3 Dataset Resampling

In this scenario, addressing the dataset imbalance, both in terms of the negative-to-positive ratio and the variations among different defects, presents a complex challenge. Attempting to achieve balance through sampling is not as straightforward as in other cases. Firstly, it is important to make some clarifications. By number of samples in a class is meant the number of patches that certainly, but not exclusively, contain the specified defect. This implies that other defects present within those patches are still considered in the pixel count. Therefore, there might be instances where there are zero samples for a given defect but not zero pixels. That being said, initially, an attempt has been made to create a training set with an equal number of samples for each class. Subsequently, a training set biased toward the 'cavity' class has been constructed. The rationale behind this approach is to assist the model in recognizing defects that are typically underrepresented, making their classification more challenging. As expected, while balancing the number of samples leads to a more uniform distribution across classes, it still results in an imbalance in the number of pixels per defect. Therefore, a dataset with an equal number of pixels per defect has been created in an effort to ensure a fair representation of all defect types.

The resampling procedure has to be custom-designed, as it relies heavily on the specific characteristics of the data being handled. This resampler focuses on oversampling each defect within an image. To construct a complete training set, these operations need to be repeated for each of the 26 images. Before sampling, a transformation is applied to the input label. In particular, the indexed label is converted into a list of One-Hot-Encoded (OHE) masks, one for each class. This transformation simplifies the pixel-wise resampling process and provides better insights throughout the entire resampling procedure. The defects have been sampled using dilation with a square kxk structuring element. Note that k has been set equal to 256, being the size of patches fed to the model. It is crucial to establish a safe area within which the defect is guaranteed to be contained. The boundaries have been defined as:

$$\begin{aligned} min &= \frac{k}{2} + 1 \\ max\_height &= H - min \\ max\_width &= W - min \end{aligned} \tag{3.3}$$

where H and W are the height and width of the image respectively. From the dilated image, only the region enclosed by the boundaries has been taken into consideration. Within the resulting cropped image, all coordinates containing the defect have been stored. Then for each pixel, a patch has been extracted from the original image centering a *"patchsize"* x *"patchsize"* square on it. The indexed label has been similarly carved out to match the patch. This procedure has been iteratively applied for each class of defects available.

To prevent the sampling of sequentially similar patches, a random sampling algorithm utilizing a uniform distribution has been introduced. Additionally, for better control over the oversampling process and to make easier various experiments, the option to **(1)** specify the proportions of samples for each class, **(2)** acquire an equal number of samples from each class and **(3)** ensure an equal distribution of pixels across classes has been implemented.

For a more straightforward visualization of the entire process, Figure 3.9 provides a schematic representation of the operations performed by the resampler on each image-label pair before they are fed into the network.

The three distinct training sets are:

- *Equal Samples*: containing the same amount of samples for each defect.
- *Oversampled Cavities*: only composed of patches sampled from the cavity class.
- *Equal Pixels*: where the patches are sampled so that the amount of pixels within each class is equal.

Regarding the *Equal Pixels* train set, note that the pixel count is not precisely the same for each defect. Instead, a tolerance of a 10% difference between the maximum and minimum pixel counts has been allowed to ensure a reasonable amount of time required for processing. Furthermore, to ensure that the model has

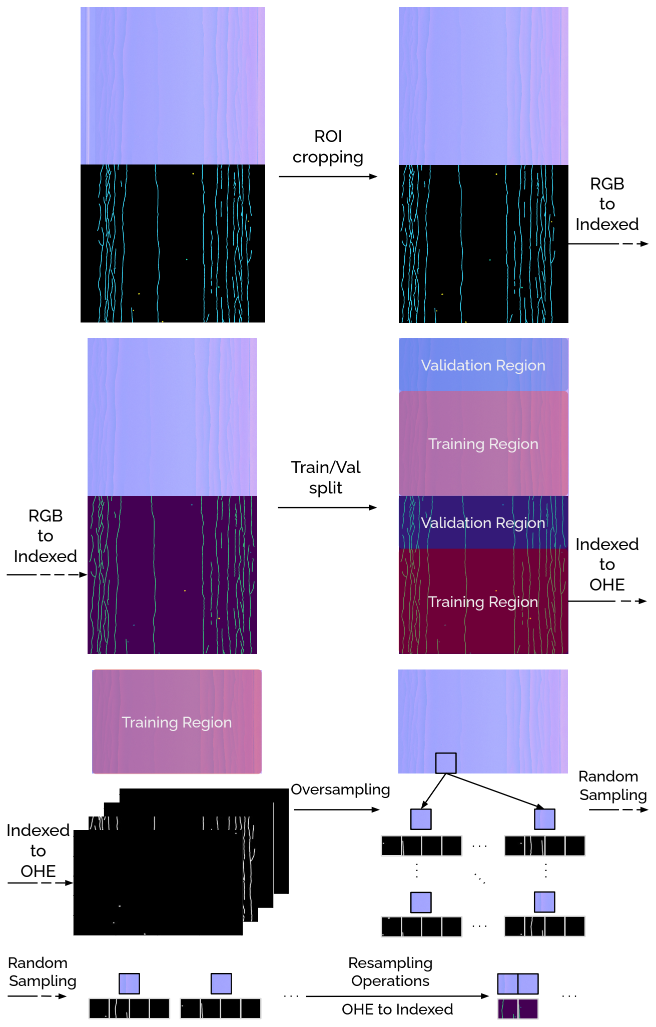


Figure 3.9: The resampling pipeline expanding the pool of possible image-label patch pair samples. The pipeline initially follows the standard sampling procedure: each original image-label pair is cropped to the Region of Interest (ROI) and the label converted from an RGB image to an indexed mask. After the split, the pipeline differs from the standard one: One-Hot-Encoded (OHE) masks for each defect are derived from the indexed one, the pool of new image and associated binary masks patches is obtained shifting the sampling window in both x and y directions. Finally, after random sampling the patches, the One-Hot-Encoded masks are joint back in an indexed one and the pairs are resampled depending on the desired train set.

the opportunity to train on a wide range of different defects, the entire training sets are resampled every *"generate_training_patches"* epochs thus getting different samples and amount of both pixels and samples each time. Note that both the *Equal Samples* and *Oversampled Cavities* train sets have almost the same amount of samples but differ in terms of pixel counts. In contrast, for the *Equal Pixels* train set, both the samples and pixel counts would vary with each resampling. Taking this into account, tables 3.3, 3.4 and 3.5 depict the typical composition of the *Equal Samples*, *Oversampled Cavities*, and *Equal Pixels* train sets, respectively.

	Negative Class	Positive Class			
	Background	Agglomeration	Cavity	Crack	Blade trail
Samples	0	351	351	351	351
Pixels	86,941,926	2,380,803	61,453	1,209,115	1,419,247
		5,070,618			

Table 3.3: Equal Samples train set composition

	Negative Class	Positive Class			
	Background	Agglomeration	Cavity	Crack	Blade trail
Samples	0	0	1518	0	0
Pixels	94,804,283	2,456,117	221,159	831,130	1,170,959
		4,679,365			

Table 3.4: Oversampled Cavities train set composition

	Negative Class	Positive Class			
	Background	Agglomeration	Cavity	Crack	Blade trail
Samples	0	31	1303	3	0
Pixels	86,691,378	240,858	218,648	231,063	239,685
		930,254			

Table 3.5: Equal Pixels train set composition

The compositions of the three train sets are notably distinct from each other. To be specific, the first train set mostly retains the same class imbalance, even though with a slight improvement due to the increased number of cavity examples. The second one greatly improves the imbalance for the cavity class with respect to both the negative class and other defects. Interestingly enough, this case highlights the proximity of cavities to agglomerations in the data. As for the *Equal Pixels* train set, it has a fair number of pixels. However, this does not necessarily guarantee that the set's composition is actually fair. Most of the samples are likely to be quite similar, which could lead to overfitting issues. Additionally, despite the imbalance between positive classes becoming irrelevant, the imbalance relative to the negative class increases considerably.

Once resampling has been explored, the focus has shifted to more adaptable, domain-invariant, methods. Working directly on CNN configurations has been the step considered the most viable.

3.4 CNN Configurations

For this study, a domain-invariant approach is considered preferable to allow the solution to be applied onto different datasets. Therefore, once the resampling strategies have been explored, the next phase involves investigating the influence of both the training procedure, through different loss functions, and the model configuration, which entails using four distinct binary classifiers instead of a single multi-class classifier. Subsequently, a combination of these two is implemented as well.

3.4.1 Binary Classifiers

Initially, a binary classifier has been employed for each defect class. The network structure has been kept the same for each model and is equal to the baseline architecture. The primary distinction lays in the training process: instead of a multi-indexed label, the network is provided with a binary mask specific to the defect it is trained to identify. Each binary classifier has been trained independently. Note that the basic loss function for the binary classifiers, Binary Cross Entropy, has been kept coherent with the multi-class classification baseline where Cross Entropy has been used.

Besides performance, the main advantage of this approach would be the possibility to control the network structure better and adapt it to suit the characteristics of each defect. This would ensure good performance while utilizing fewer computational resources. However, this possibility has not been explored, as the aim is to stay relatively close to the baseline. So, while the computational cost per model has remained the same as that of the baseline, since a separate model has been trained for each defect, the overall cost is approximately four times that of the original. A better insight into the computational costs is given in Table B1 and Table B2.

Following the initial binary baseline group of models, another group has been trained using a modified train set in which the specific class is present in 50% of the samples. The data feeding pipeline closely resembles the baseline approach, with one significant distinction: only the binary mask corresponding to the defect the model has been trained on is used as the label, as depicted in Figure 3.10.

During the inference phase, each model is supplied with the same input image, as shown in Figure 3.11.

3.4.2 Loss Change

The second approach has involved testing different loss functions in place of the standard Cross Entropy (CE) Loss [24]. Both Weighted Cross Entropy (wCE) Loss and Focal Loss (FL) have been explored with different weightings applied. To calculate these weightings, two different sets of class weights have been derived using Inverse Class Frequency (ICF) and Inverse Square root Class Frequency (ISCF), which have been computed based on the training regions of each image. The formulas for calculating ICF and ISCF are as follows:

$$\begin{aligned} ICF &= \frac{1}{n\_samples_c} \\ ISCF &= \frac{1}{\sqrt{n\_samples_c}} \end{aligned} \tag{3.4}$$

where $n\_samples_c$ is the number of samples in each class.

Each class weight has then been normalized by the number of classes:

$$\alpha_c = \frac{f_c}{\sum_k f_k \;\; * \text{n\_classes}} \tag{3.5}$$

where $n\_classes$ is the number of classes, 5 in this case. The values computed according to both weighting schemes on the training regions of each image are detailed in Table 3.6.

To gain a more detailed understanding of the distinct effects of simple class weighting and altering the loss function, these two strategies have been tested independently. Weighted Cross Entropy is defined as follows:

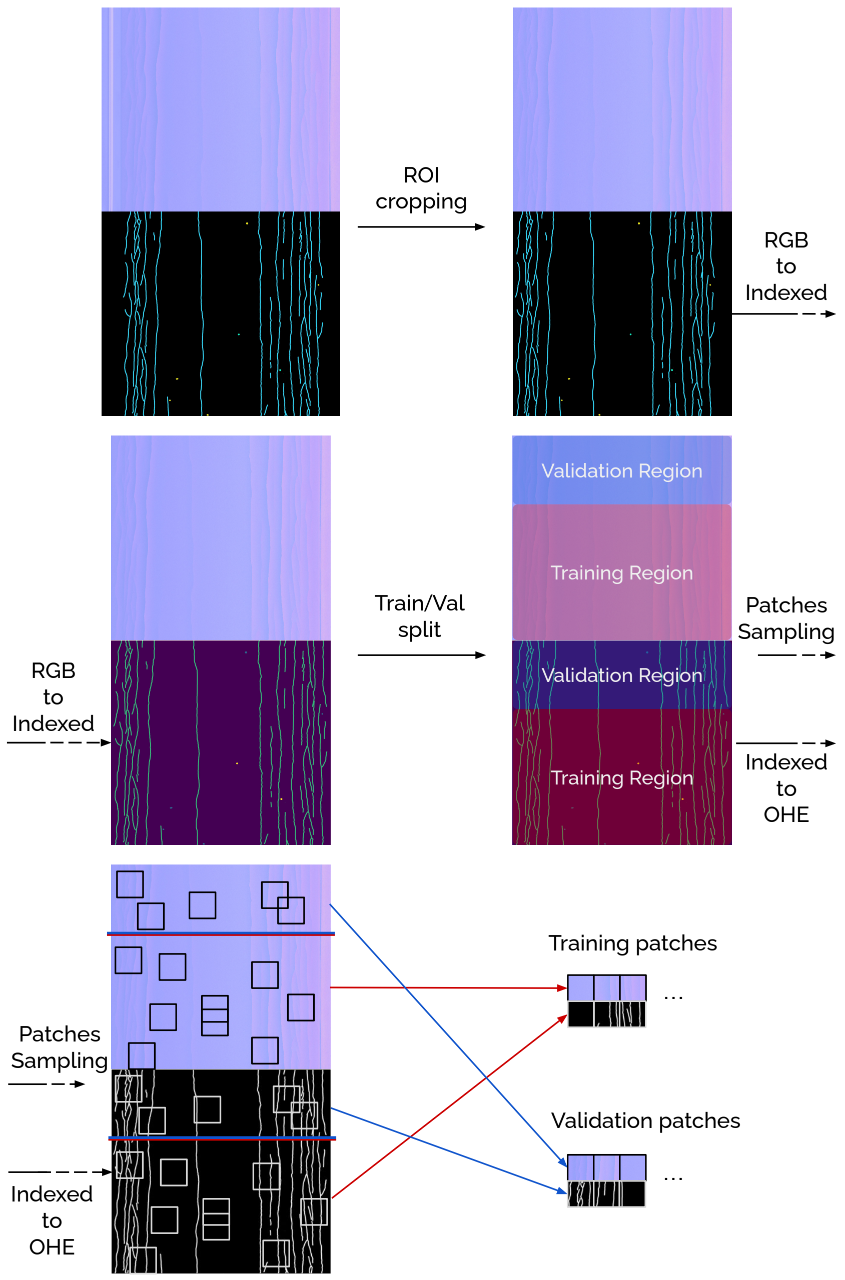


Figure 3.10: The binary sampling pipeline providing image-binary mask pair samples to the network. After cropping each image-label pair to the Region of Interest (ROI), the RGB-to-Indexed transformation is applied to the label. Following the train/validation region split, the binary mask relative to the defect of interest is derived from the indexed one. Finally, the patches pairs are randomly sampled from the train or validation region depending on the purpose.

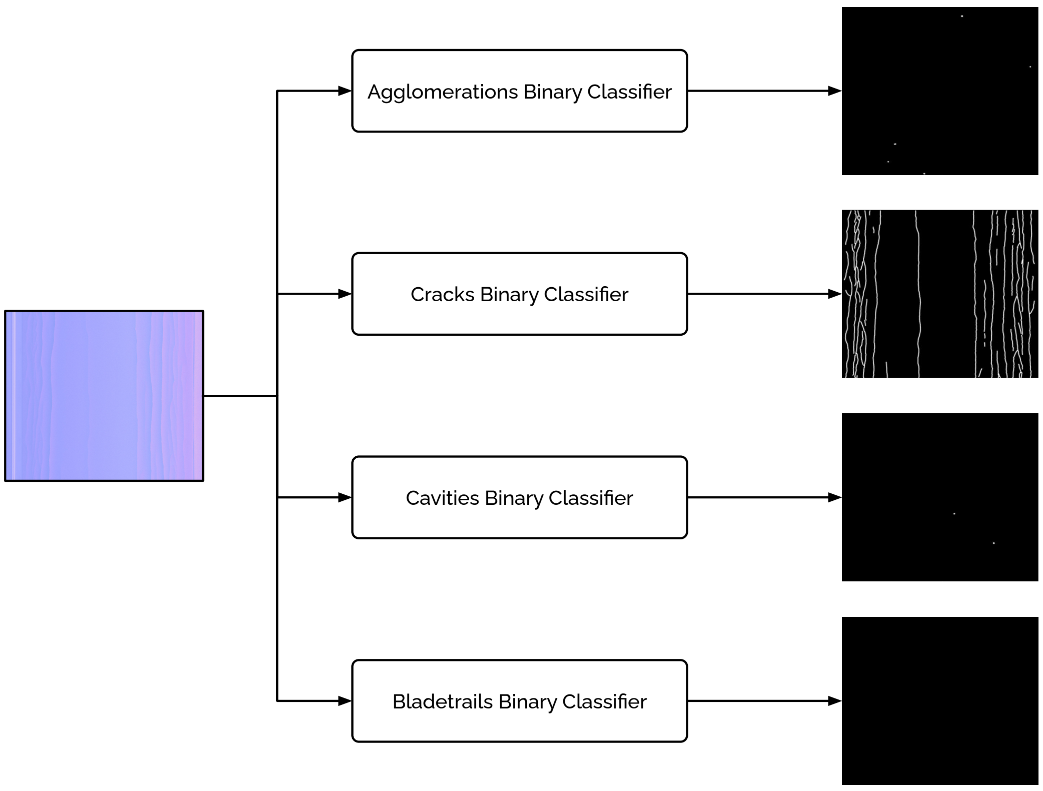

Agglomerations Binary Classifier
Cracks Binary Classifier
Cavities Binary Classifier
Bladetrails Binary Classifier

Figure 3.11: Binary classifiers inferring. The same input image is fed to the four binary models to compare their performance with the multi-class models.

	Negative Class	Positive Class			
	Background	Agglomeration	Cavity	Crack	Blade trail
ICF α	0.003	0.111	4.355	0.309	0.226
ISCF α	0.075	0.475	2.978	0.793	0.678

Table 3.6: Class weights used in Weighted Cross Entropy and Focal Loss. The weights have been computed using Inverse Class Frequency and Inverse Square root Class Frequency.

Given

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases} \qquad \alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1 - \alpha & \text{otherwise} \end{cases} \tag{3.6}$$

then

$$wCE(p_t) = -\,\alpha_t \log p_t \tag{3.7}$$

In the above, $y \in \pm 1$ specifies the ground-truth class and $p \in [0, 1]$ is the model's estimated probability $y = 1$. This basic definition, which refers to the binary case, has then been adapted to the multi-class one using the computed values shown in Table 3.6.

Regarding the Focal loss, besides the two weighting schemes previously discussed, a model has been trained using the same weightings specified in the original paper [13], namely 0.25 for the negative class and 0.75 for the positive class, which, in this particular case, includes all defects. Focal Loss is defined as follows:

$$FL(p_t) = -\,\alpha_t\,(1 - p_t)^{\gamma}\,\log p_t \tag{3.8}$$

As for wCE, this has later been adapted to the multi-class scenario. Originally designed for object detection, the Focal Loss has often been adopted to solve semantic segmentation [7] [4]. It mainly consists in a variation of the CE loss function, reshaping it to down-weight easily classified examples, thus focusing learning on challenging negatives. This is achieved through a modulating factor

$(1 - p_t)^\gamma$ with tunable focusing parameter $\gamma \geq 0$.

The authors [13] explored several values of $\gamma \in [0, 5]$ and observed two key properties of the Focal Loss:

1. *"When an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As $p_t \to 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted."*

2. *"The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When γ = 0, FL is equivalent to CE, and as γ is increased the effect of the modulating factor is likewise increased"*

Thus the loss contribution given by easily classified examples gets reduced by the modulating factor. Besides, it increases the range in which an example would be considered as easy, thus receiving low loss and increasing the importance of correcting misclassified examples. This indicates the Focal Loss' capability to deal with intra-class imbalance which, joint with class weights mitigating the effect of inter-class imbalance, helps the model to ease imbalance during training.

Authors [13] found γ = 2 to work best in their experiments. Observing the results obtained in the first trials using Focal Loss, γ = 2 has been considered appropriate in our case too, as the influence of the modulating factor is still reasonable within that value.

3.4.3 Binary Focal Loss Classifiers

Following the results obtained, which will be discussed in the next chapter, the final step has involved combining both the One-vs-Rest approach and loss change strategies. For the One-vs-Rest approach, only standard binary classifiers, without resampling, have been utilized since they have been considered the more suitable approach. So, a first base group of binary classifiers has been trained using focal loss with standard weighting, as in the previous multi-class focal loss implementation. Then, the weights considered the best have been utilized, in this case using ISCF as the weighting scheme.

Given that the negative/positive class compositions vary for each model, the weights using ISCF have been recalculated. Specifically, the negative class has

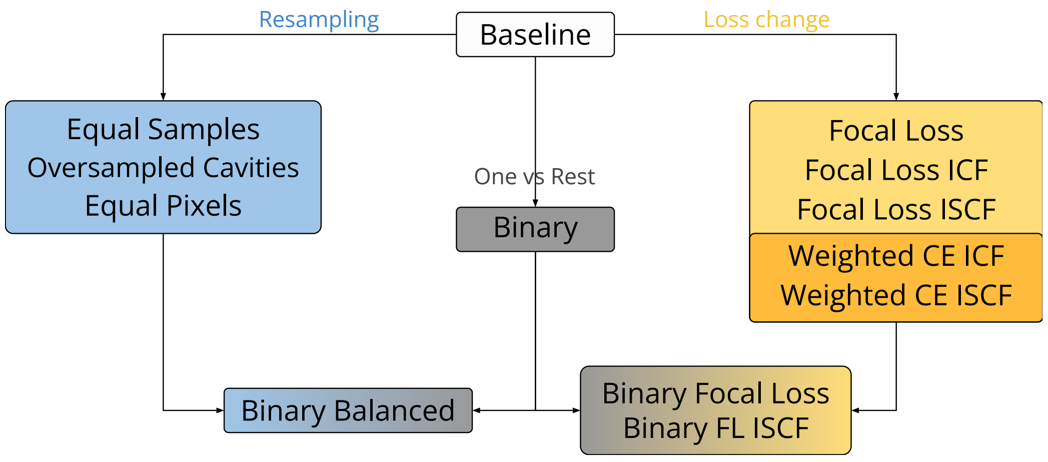


Figure 3.12: Models summary.

been defined as comprising all classes except the defect the model is trained to detect. Table 3.7 shows the recalculated weights.

	Model			
Class	Agglomeration	Cavity	Crack	Blade trail
Negative	0.135	0.085	0.024	0.098
Positive	0.865	0.915	0.976	0.902

Table 3.7: Class weights used in Focal Loss for binary models. The weights have been computed using Inverse Square root Class Frequency.

Before delving into the evaluation process and its outcomes, providing a summary of the implemented models, as depicted in Figure 3.12, will prove beneficial to the reader.

3.5 Evaluation

The entire evaluation process entails a series of decisions and precautions, primarily due to the dataset's inherent imbalance. In fact, as will be discussed later on, due to some uncertainties regarding numeric values, the dataset's nature, and the task at hand, a quantitative analysis is not sufficient to evaluate the performance. A qualitative analysis is essential in this case study.

The performance analysis of the models is performed on the validation region of each image. Normally, the model would take patches as input and return the corresponding predicted label patches as output. So, to evaluate the whole validation region, it should be fed patch-wise to the network and its predicted label reconstructed by stitching together the individual patches in output. Unfortunately, due to the padding present in convolutional layers, which produces artifacts at the borders of each patch, this would lead to an increase in noise generation. A simple and quick solution is to feed the whole image region instead. This is possible due to the fully convolutional nature of the UNet. By doing so, also the reconstruction of the label through the repositioning of individual inferred patches is bypassed.

3.5.1 Quantitative Analysis Method

First of all, for each model and for each defect, a set of basic pixel-wise metrics has been calculated on all images, namely: Positives, Negatives, True Positives, True Negatives, False Positives, and False Negatives. Furthermore, TPR, FPR, and Precision have been computed as micro-averages from basic metrics. Note that, given the possibility of basic metrics being zero, ratios have been computed only where feasible. This has often been the case for Precision.

For each model, two performance comparisons have been conducted:

- TPR/FPR comparison
- Precision/Recall comparison

TPR holds significant importance in this case as it measures the model's ability to detect defects. At the same time, FPR is a critical consideration as it indicates how often the model might erroneously detect false defects. Detecting a defect in an image that does not contain any could result in halting production and unnecessary manual inspection of the foil, incurring avoidable expenses. TPR/FPR comparison is considered more informative compared to the Precision/Recall comparison as it makes it immediate to evaluate the trade-off between the cost of false positives and false negatives already mentioned in Chapter 2.3. All the metrics used must still be interpreted with caution in this case due to the dataset's imbalance.

The metrics have been computed for each defect and then averaged, enabling both generic and defect-specific evaluations. To better compare models and visualize trends, two charts illustrating the two performance comparisons have been plotted. Furthermore, plots displaying together the performance on individual defects and their average on each metric per model have been generated. These visual representations prove valuable in assessing whether a model exhibits consistent performance across all classes or displays variations. These plots can be found in Section 4 for reference.

For the main variants of the baseline model, the following additional tests have been carried out:

- Performance Stability over (later) Iterations (PSoI)
- Performance Stability over Multiple Training (PSoMT)

These tests have been conducted to gain deeper insights into the stability and reliability of each model. In the PSoI analysis, the objective is to determine if the model has reached convergence in its later iterations. This is achieved by inferring the model and evaluating it using different checkpoints, specifically at intervals of 500 iterations, commencing from iteration 7000. To provide a precise measure of the model's consistency, the standard deviation of the models' checkpoint averages has been computed for each metric used in the main comparisons. On the other hand, the PSoMT tests are essential to assess the influence of different kinds of random sampling on the training. Each model has been trained three times, and the same statistical analysis that has been conducted for PSoI has been replicated for PSoMT. The results have been assessed individually, and standard deviations have been considered acceptable depending on the models' performance.

3.5.2 Qualitative Analysis Method

Considering the unique nature of the dataset already discussed in section 3.1, a qualitative analysis is crucial to ascertain the reliability of the quantitative analysis and to gain deeper insights. This qualitative analysis has proved invaluable in understanding why certain metrics are better or worse in some models and whether those variations are truly indicative of the model's performance. To conduct this analysis, each model has been inferred on the validation region of every image, and the resulting predictions have been plotted. Additionally, to help the evaluation, a three-way comparison chart containing the input validation region, the original annotations, and the predicted ones has been constructed. Three key evaluations have been performed on this:

- Predictions and original annotations comparison
- Annotation errors identification
- Predictions and input image comparison

A selection of the most representative or distinctive images, shown in Figure 3.13, has been chosen for a more in-depth analysis. For each of these selected images, an exhaustive inspection and comparison of the predictions inferred by the different models has been meticulously conducted.

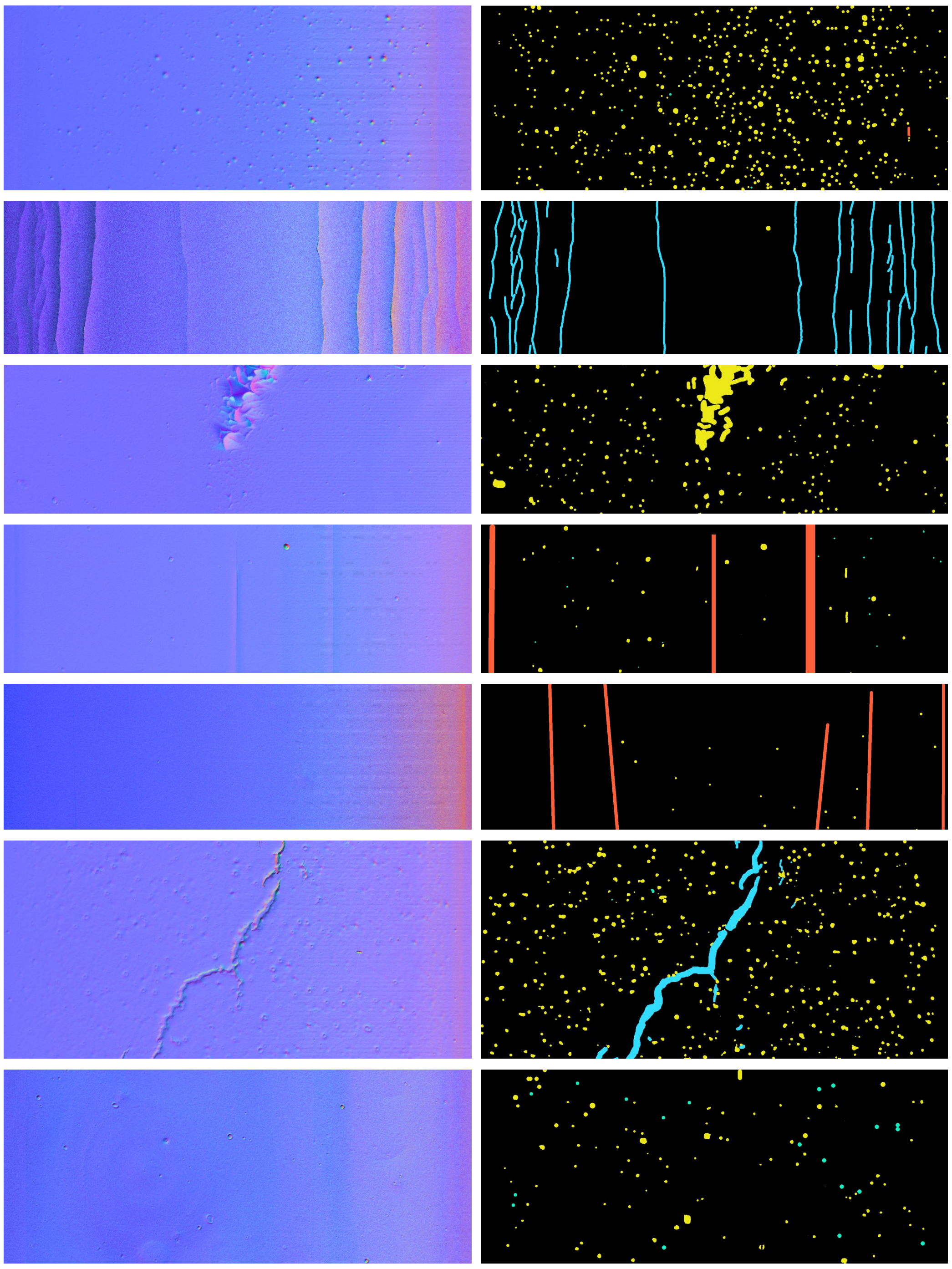


Figure 3.13: Validation regions **(left)**, and respective annotations **(right)**, chosen for manual in-depth analysis. All defects are represented as well as some peculiar cases.

Chapter 4

Results

In this section, we examine the results obtained through the evaluation process outlined in the previous section. These results are presented and analyzed in isolation from certain key observations, which will be addressed in the subsequent section. This separation is intended to delineate the actual accomplishments from potential or hypothetical scenarios.

4.1 Quantitative Analysis Results

As said previously, each model's first observed results have been a set of basic metrics computed for each defect. These are reported in the appendix in Table A1.

The TPR/FPR and Precision/Recall comparisons have been the first proper step of the evaluation process. Figure 4.1 shows the plots that have been constructed and analyzed for this purpose. Specifically, in the TPR/FPR comparison almost every model outperforms the baseline in terms of TPR. The exception is the Equal Pixels model, which exhibits markedly lower performance compared to the baseline but exhibits the lowest FPR. Notably, the wCE ICF and FL ICF models achieve peak TPR while having the worst FPR. The remaining models exhibit a trade-off between TPR and FPR, with Oversampled Cavities and FL ISCF showing promising TPR/FPR balances. In the Precision/Recall performance comparison, focusing on Precision, only the Equal Pixels, Oversampled Cavities, and Binary Balanced models outperform the baseline. Conversely, the wCE ICF and FL ICF

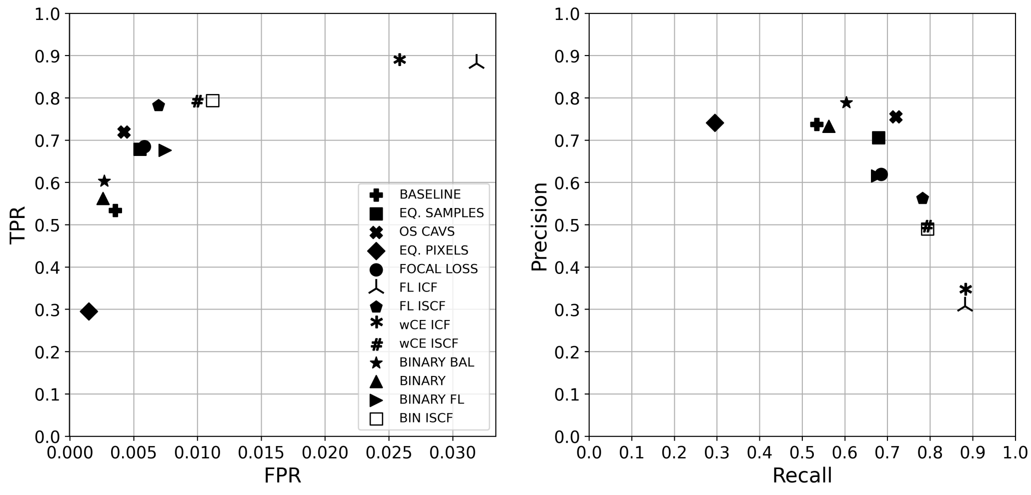


Figure 4.1: TPR/FPR and Precision/Recall comparisons. **Left:** Black shapes indicate the performance (TPR/FPR) of the respective models shown in the legend and explained in the previous chapter. **Right:** Black shapes indicate the performance (Precision/Recall) of the same models.

models perform the poorest in terms of Precision. The remaining models prove a trend where improved TPR typically corresponds to reduced Precision. Notably, Oversampled Cavities is the model with the most favorable Precision/Recall trade-off, as revealed in the plot.

The variation plots, depicted in Figure 4.2, show how stable the models' performance is on the different classes. The TPR Variation plot highlights how cavities exhibit the poorest performance in the baseline model, significantly affecting the overall average, while on agglomerations the model performs exceptionally well. This pattern is largely consistent across most models, with Binary, Equal Pixels, and Binary Focal Loss models displaying higher variability, while Oversampled Cavities and models utilizing loss change exhibit lower variance. In the FPR Variation plot, the baseline model under-performs on agglomerations, while on cavities it shows remarkable performance. This trend is consistent across most models, with FL ICF and wCE ICF models displaying higher variability, whereas Equal Pixels and Binary models exhibit lower variance. FPR shows a wide range of fluctuations, which lessen as average FPR decreases. For very high variance cases, agglomerations and blade trails remarkably raise the average. Note how FPR values are always quite small suggesting a potential lack of significance. More on

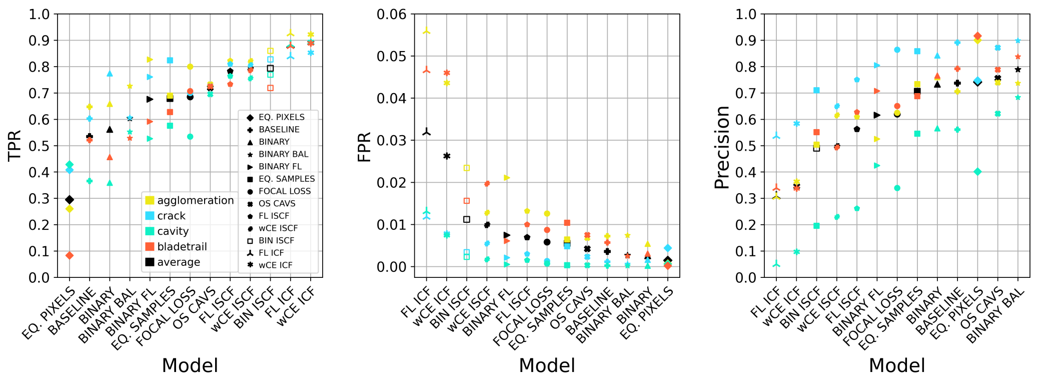


Figure 4.2: Variation Analysis of TPR **(left)**, FPR **(center)** and Precision **(right)**. Different shapes indicate the performance of the respective models shown in the legend and explained in the previous chapter. Different colors of the same shape indicate the performance on respective defects shown in the legend. Black shapes indicate the models' performance averaged over single defects.

this will be said in the next chapter. The Precision Variation plot shows how cavities are the weakest performers in the baseline model, while cracks achieve the highest precision. This trend is mirrored across most models, with higher variance observed in Equal Pixels and models employing Focal Loss, and lower variance in Binary Balanced and Oversampled Cavities models.

The last quantitative assessment step have been the PSoI and PSoMT tests. Table 4.1 reports the values computed from both the resilience tests for each model. In every case, the standard deviations are low enough (< 0.05) to consider the models' performance as stable in the latest iterations, meaning that models have converged after 10.000 iterations, and on multiple training. This also shows that randomness does not significantly influence the training procedure.

Quantitatively, Oversampled Cavities appears to be the top-performing model, demonstrating a strong trade-off in both performance comparisons while maintaining low variation. Conversely, wCE ICF, FL ICF, and Equal Pixels are expected to perform less effectively. As shown in the next section, this is not exactly the case. To validate the quantitative findings and gain insights into the underlying reasons for specific results, a qualitative analysis becomes necessary.

Model	PSoI				PSoMT			
	TPR SD	FPR SD	Pr. SD		TPR SD	FPR SD	Pr. SD	
Baseline	0.01	0.0002	0.011	✓	0.011	0.0009	0.025	✓
Os. Cavities	0.013	0.0004	0.018	✓	0.014	0.0008	0.022	✓
Binary	0.068*	0.0015	0.035	✓*	0.003	0.0001	0.008	✓
wCE ICF	0.026	0.007	0.032	✓	0.008	0.002	0.013	✓
Focal Loss	0.036	0.001	0.028	✓	0.043	0.0005	0.014	✓
Binary FL	0.021	0.0006	0.016	✓	0.009	0.0003	0.003	✓

* Value significantly affected by the earliest analyzed iteration. The test is still considered as passed for this reason.

Table 4.1: Performance Stability over (later) Iterations (PSoI) and Performance Stability over Multiple Training (PSoMT) resilience tests results. For each tested model, the standard deviation (SD) for each metric is reported.

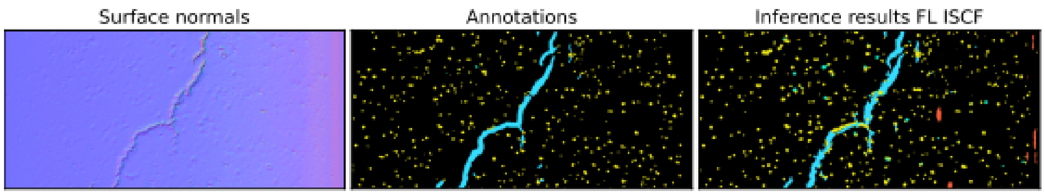


Figure 4.3: Multi-class models' comparison example.

4.2 Qualitative Analysis Results

However important the numbers may be considered, the actual results are given by the segmented images. So, starting from the results obtained in quantitative analysis, the best and worst models have been investigated thoroughly. Qualitative analysis has changed the stance towards some of these models' performance assigning meaningfulness to those numbers and shifting the perspective on overall results.

Figure 4.3 and Figure 4.4 present a sample comparison between labels and predictions in the multi-class and binary cases, respectively, while Figure 4.5 shows the predictions made by each model on the same patches, where the baseline model failed, that have been shown in Figure 3.8 of section 3.2. Specifically, in the first case of Figure 4.5, the missed blade trail is detected at least partially by all models, except for the baseline, with FL ISCF and BIN ISCF showing the best results. In the second case, many models miss the non-annotated cavities as the baseline. Only ICF models detect all the interested cavities but overestimate their size and generate noise at the same time. So the results considered as most satisfactory are given once more by FL ISCF and BIN ISCF.

The results shown in Figure 4.5 along with the performed manual in-depth inspection highlighted patterns in models' results. Among the models within the resampling group, Oversampled Cavities and Equal Pixels have yielded interesting results. Despite expectations, Oversampled Cavities does not perform as well as expected even with respect to the baseline model. Although it improves defect detection, particularly cavities, it often generates more noise. On the other hand, the Equal Pixels model, while producing less noise, overlooks most defects, resulting in mediocre overall performance.

The Binary model produces outcomes akin to the baseline, exhibiting improve-

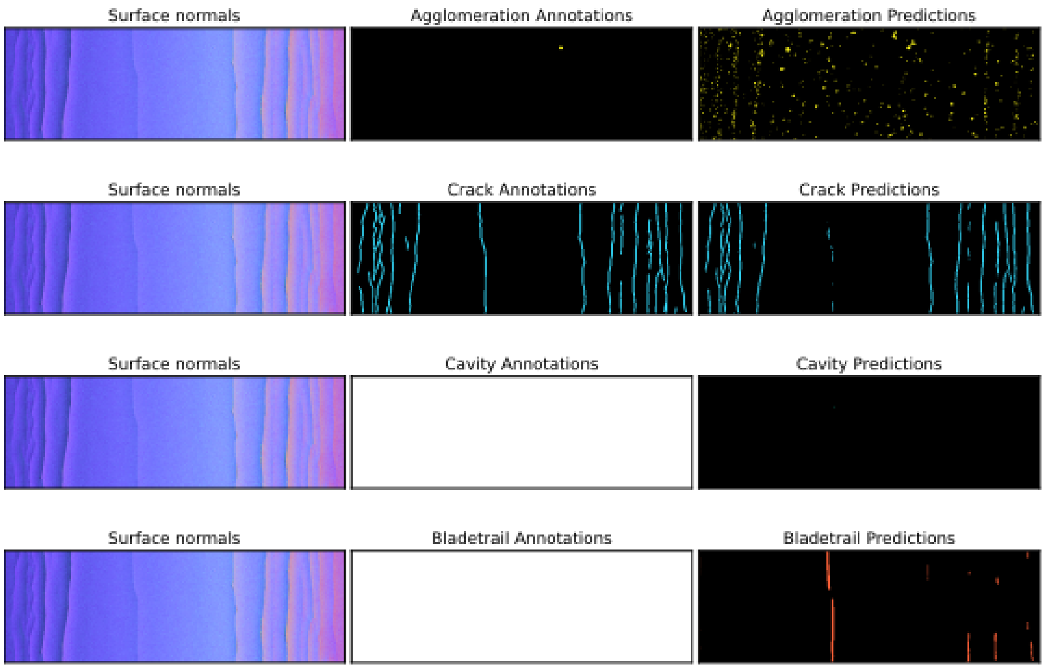


Figure 4.4: Binary models' comparison example. The plot shows results inferred by the Binary Focal Loss model. Empty annotations are represented as a completely white mask.

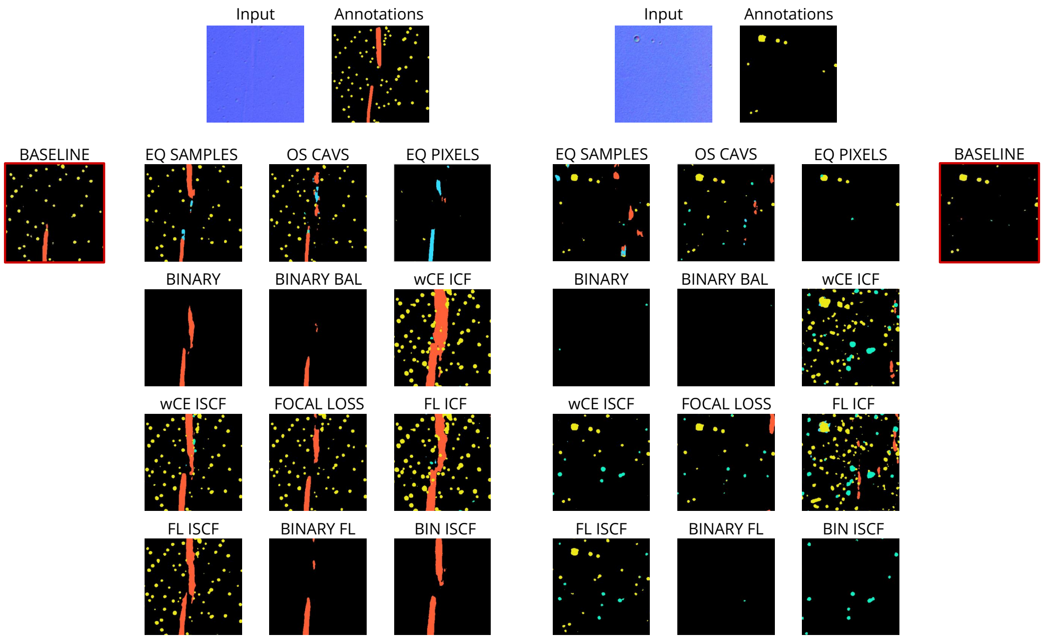


Figure 4.5: Predictions by models on the patches where the baseline fails. In the first case, the baseline misses an annotated blade trail. In the second case, the baseline misses some cavities which are not even annotated. **Note:** for binary models only predictions of interested defects are shown to ease the comparison.

ments with cracks but deteriorations with blade trails. It usually generates less noise. Meanwhile, the Binary Balanced variant does not significantly enhance predictions and occasionally yields worse results by ignoring some defects, leading to an increased allocation of pixels to the background. This accounts for the higher precision observed in quantitative analysis.

Models employing Focal Loss or weighted Cross Entropy loss tend to overestimate the size of defects rather than misclassify or generate noise. This explains the high FPR values that have been obtained in the quantitative analysis and should be taken into account when evaluating the performance. Specifically, ICF weighting steers the training too much, causing the model to exasperatedly overestimate defect sizes, detect more defects than any other model, but generate noise at the same time. All this results in wCE ICF and FL ICF performing poorly, according to the quantitative analysis. On the other hand, ISCF achieves a good trade-off between defect size and detection while maintaining a clean resulting image. These observations also apply to Binary FL and Binary FL ISCF variations.

Summing up the results:

- Resampling approaches don’t align with the quantitative analysis
- Binary models don’t show significant improvement over the baseline
- Loss change approaches require caution but can yield satisfactory results with proper weighting
- Binary models using focal loss do not really improve the results already obtained by the multi-class ones

Chapter 5

Discussion

In addition to the results presented in the previous chapter, in this section some key observations and clarifications, that allow for a deeper understanding of the work, are presented. Therefore this section is fundamental to contextualize the results obtained and the choices made during the design phase. Specifically, the impact of each component, from dataset to imbalance mitigation strategies, on the models' performance is analyzed. Also, some considerations on the evaluation procedure and its reliability are made.

First of all, some considerations on the dataset. The manual annotation procedure on large images can be very time-consuming and difficult, especially if a high accuracy is desired. Indeed, the defects' labeling process has not been fine-grained and presents some annotation errors usually given by coarse annotations or overlooking small defects. This can negatively influence the training process. Furthermore, defects that are not labeled correctly in the validation set but are properly spotted by the model will contribute to metrics miscalculations. Specifically, those pixels will be counted as False Positives instead of True Positives, affecting all the metrics used in quantitative comparisons. This needs to be considered during the models' performance evaluation, especially in quantitative analysis. In this study, it has been addressed evaluating the results also in a qualitative manner, highlighting when metrics don't do justice to actual results. Figure 5.1 shows an example of coarse annotation where the annotated area for a blade trail takes many more pixels than its actual size. Figure 5.2 gives an insight on the amount of missing

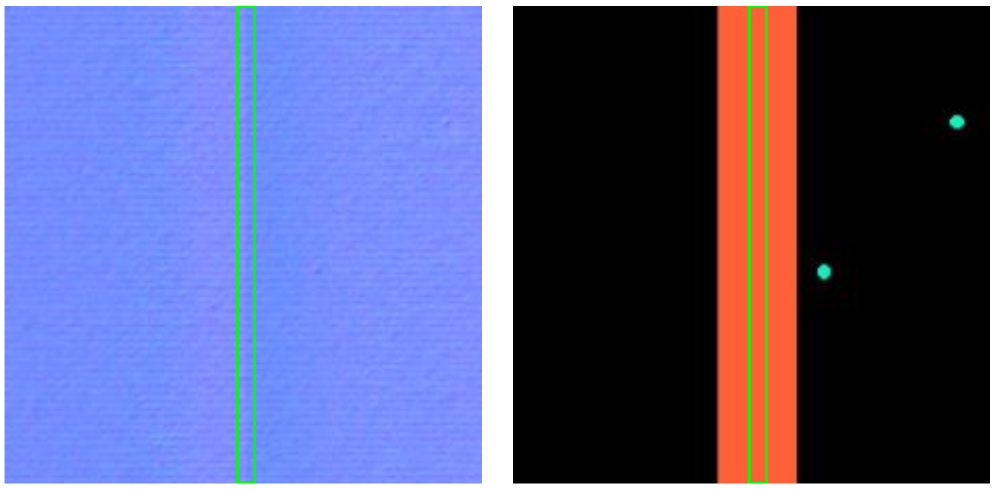
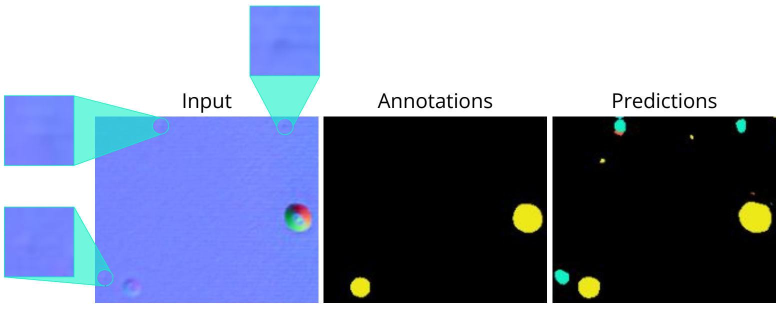


Figure 5.1: Example of coarse annotation (orange) compared to the actual size of the defect (green box).

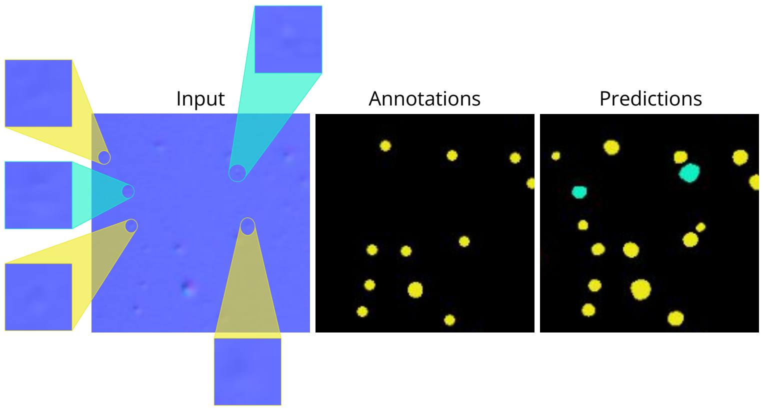
annotations. In fact, the picture shows three separate examples where many cavities and agglomerations have not been annotated. Moreover, Figure 5.3 shows an example of barely visible defects that are only spotted by models, highlighting the low reliability of quantitative metrics and the complexity of the manual annotation process.

Implementing resampling has encountered various challenges, questioning its worthiness. It is a domain-specific task that necessitates a distinct approach for each dataset and, while the fundamental concepts might be analogous across datasets, the actual implementation varies significantly. In this specific case, the intrinsic complexity of the dataset adds further complexity to the resampling process. Much effort is required to navigate data structures and types, particularly if the dataset hasn't been structured with resampling considerations in place. Another critical consideration revolves around achieving a balanced dataset. As previously mentioned in section 3.3, creating a dataset with an equal number of defect samples and subsequently with the same number of pixels per defect has been undertaken. While the former remains feasible within reasonable time and computational constraints, the latter is both difficult to implement and very expensive in terms of time and computational cost but is needed to get a fair dataset numerically.

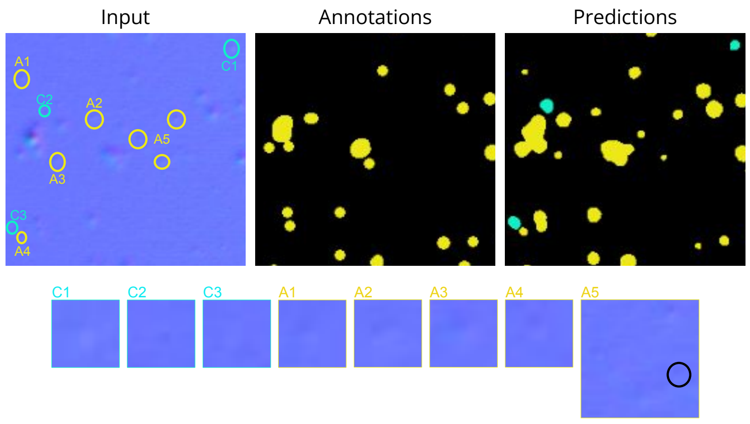
As expected and observed in model evaluation within section 4, oversampling this kind of dataset leads to overfitting in most of the cases. Indeed, despite the



(a)



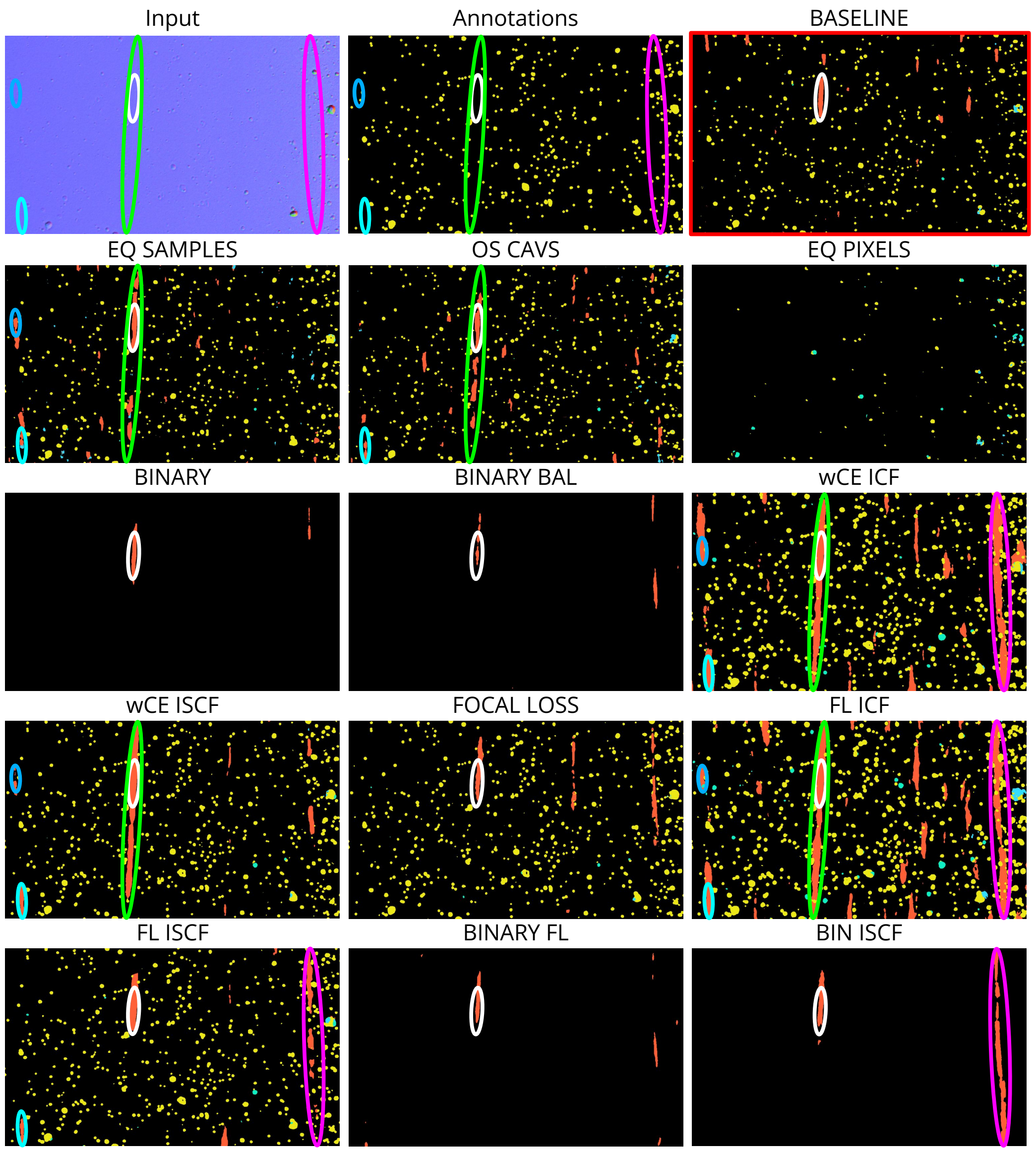
(b)



(c)

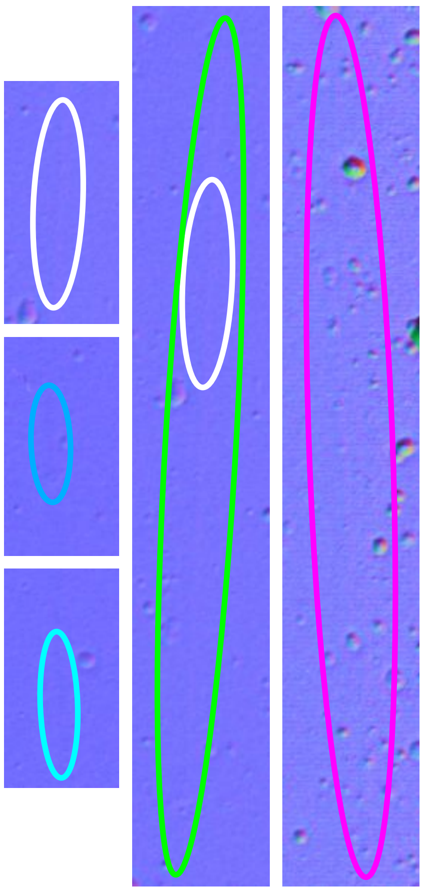
Figure 5.2: Examples of missing annotations. In **(c)** the black circle in A5 highlights an agglomeration which is neither annotated nor detected by the model.

increase in the volume of examples provided to the network, the homogeneity of defects has reduced the diversity within the samples. Therefore, the network's abil-



(a) The blade trails that have been clearly identified by several models are highlighted in predictions, original annotations and input image. These defects, although possibly inconspicuous to annotators who overlooked them, gain credibility as different models consistently detect them, indicating their actual presence in the image.

Figure 5.3: Example of spotted-by-models-only annotations.


(b) Detail of the detected blade trails in the input image.

Example of spotted-by-models-only annotations.

ity to effectively generalize across different types of defect is hampered, causing overfitting. This trend is particularly noticeable within the 'cavity' class, significantly underrepresented in the dataset. Additionally, in the case of the equal pixels resampled train set, the repetitive sampling of same patches to match the pixel counts of other defects has further compounded this issue. Using resampling, data variance is impossible to grasp in this case. By Equal Pixels one might have anticipated better results concerning the baseline or other models using resampling. However, as explained before, this is not the case given the lowering variance of samples due to resampling a smaller pool of patches while striving the number of pixels toward an equal amount per defect. This situation has led to overfitting. It could be argued that by using techniques such as early stopping or hoping for better pools of patches during the training process, it could have performed better. Nonetheless, the model has not been retrained due to different reasons:

- The time required to find optimal settings would not have been worth it as it ranged between 50 and 70 minutes.
- Increased energy consumption due to the generation of additional pools of patches would have significantly affected the training costs.
- It has proved that resampling strictly depends on data and it is impossible to make the dataset fair in this case.

Resampling surely may represent a valid option for imbalanced datasets but is not an effective solution per se as it leads to overfitting if no countermeasures are carefully taken, as the results in this study show. Furthermore, due to different problems arising during implementation, the marginal improvement in results is not worth the effort in this case as the performance improvement has been sensible only when over sampling the minority class. In that case, quantitative analysis shows an overall improvement with respect to the baseline model: $+0.186(+35\%)$ in TPR, $+0.018(+2.5\%)$ in precision but a $+0.0006(+17\%)$ in FPR as can be seen in Figure 4.1. The qualitative analysis has showed improvements but has not matched the expectations based on quantitative analysis. Anyway, this solution remains costly and performs worse than quick and generalizable loss changing solutions. Nonetheless, if thoughtfully integrated with other solutions, resampling strategies might be viable.

Some other obtained results also need further explanation, in particular for those models from which one might have expected more. One scenario where the expectations have fallen short is that of binary models. Considering the basic Binary model, the improvement on performance, shown in Figure 4.1, is marginal on TPR, as only a $+0.028(+5\%)$ is observed, consistent on FPR with a $-0.001(-28\%)$, unchanged on precision and no significant enhancements are visible in qualitative analysis. The slight change in performance over the baseline does not justify quadrupling the time and resources required. While adapting the network architecture to accommodate each defect could potentially be a solution worth exploring, this type of overhaul is not within the scope of this thesis. The same holds for variants hybridized with resampling or focal loss.

Regarding ICF-weighted models, some observations need to be made as well. Initially, considering how imbalanced the dataset is, Inverse Class Frequency has been thought of as an appropriate weighting scheme: high striving for high imbalance. Then, the quantitative analysis has given a first insight into how much it influences the learning process and the qualitative analysis has provided further proof that ICF influences the learning process too heavily making the network predict oversized defects, thus getting an overall imprecise background as well. While this weighting scheme might be effective for other datasets or tasks, a good way to improve results while also being cautious is to apply lighter weighting, as ISCF in this case. In fact, although the quantitative improvements concern exclusively the TPR, marking a notable $+0.248(+47\%)$ with respect to the baseline, while the remaining metrics, FPR and Precision, worsen, marking a $+92\%$ and -24% respectively, the qualitative analysis actually shows the best results obtained. Note how these results are obtained at a low cost as, once the focal loss is implemented, the only needed step is to compute the class weights, which can be automatized when relying on statistics as in this case.

Chapter 6

Conclusions and Outlook

This thesis has explored the problem of severe class imbalance in semantic segmentation of industrial images of battery foils. Various methods have been proposed and evaluated to mitigate the effects of class imbalance on the performance of convolutional neural networks. These methods include resampling techniques, loss function modifications, and the One-vs-Rest approach. The results have shown that some of these methods can improve the segmentation accuracy and robustness, especially for the minority classes. However, the results have also revealed some limitations and challenges of the methods, such as the trade-offs between sensitivity and specificity and the computational cost. Furthermore, the results have demonstrated the importance of conducting both quantitative and qualitative analyses to assess the performance of the models on imbalanced datasets, as some metrics may not reflect the true quality of the segmentation.

This work has contributed to the field of semantic segmentation by providing a comprehensive understanding of the problem of severe class imbalance and offering some effective strategies to address it. The work has also highlighted some directions for future research, such as exploring synthetic data generation techniques, to enhance the performance of semantic segmentation models on imbalanced datasets.

From a practical perspective, the work also has implications for battery foil manufacturing and quality control processes, as it can help to detect and classify defects more accurately and efficiently. This can lead to improved product quality, reduced waste, and lower costs. The methods developed in this work can also be applied to other types of industrial images or imbalanced datasets, such as medical images, satellite images, or face recognition.

However, the work also has some limitations that need to be addressed in future research. One limitation is that the methods may not generalize well to new or unseen data, as they may overfit the training data or suffer from domain shift. A possible way to overcome these limitations is to use active learning [23] or semi-supervised learning techniques [14], which can reduce the need for manual annotation and leverage unlabeled data to improve the model performance. Another possible way is to use transfer learning [19] or meta-learning techniques [2], which can enable the model to adapt to new domains or tasks with minimal data or supervision. These techniques could enhance the robustness and applicability of the methods for semantic segmentation of imbalanced datasets. Additionally, deeper hyperparameter tuning could help to further improve the performance of the models. Another possibility could be a confidence analysis which could help to measure the uncertainty of the models and identify the cases where they are more likely to make mistakes. These techniques could improve the reliability and interpretability of the models.

In conclusion, this thesis underlines the importance of addressing class imbalance in semantic segmentation and proposes some methods to counteract it while emphasizing the importance of a proper evaluation process in similar cases.

Appendix A

Metrics results

Model	Defect	P	TP	FP	N	TN	FN
Baseline	Agglomeration	750,144	485,695	220,976	30,877,456	30,656,480	264,449
	Cavity	17,654	6,455	6,473	31,609,946	31,603,473	11,199
	Crack	279,658	168,512	34,049	31,347,942	31,313,893	111,146
	Blade trail	365,018	190,104	177,654	31,262,582	31,084,928	174,914
Eq. Samples	Agglomeration	750,144	517,101	197,715	30,877,456	30,679,741	233,043
	Cavity	17,654	10,156	10,947	31,609,946	31,598,999	7,498
	Crack	279,658	230,349	150,995	31,347,942	31,196,947	49,309
	Blade trail	365,018	229,055	324,958	31,262,582	30,937,624	135,963
Os. Cavities	Agglomeration	750,144	549,521	208,790	30,877,456	30,668,666	200,623
	Cavity	17,654	12,277	10,025	31,609,946	31,599,921	5,377
	Crack	279,658	202,850	71,651	31,347,942	31,276,291	76,808
	Blade trail	365,018	264,613	235,795	31,262,582	31,026,787	100,405
Eq. Pixels	Agglomeration	750,144	195,305	22,002	30,877,456	30,855,454	554,839
	Cavity	17,654	7,565	19,063	31,609,946	31,590,883	10,089
	Crack	279,658	114,044	136,827	31,347,942	31,211,115	165,614
	Blade trail	365,018	30,516	5,550	31,262,582	31,257,032	334,502
Binary	Agglomeration	750,144	494,236	165,726	30,877,456	30,711,730	255,908
	Cavity	17,654	6,339	5,637	31,609,946	31,604,309	11,315
	Crack	279,658	216,511	50,717	31,347,942	31,297,225	63,147
	Blade trail	365,018	166,635	98,195	31,262,582	31,164,387	198,383
Bin. Balanced	Agglomeration	750,144	544,595	228,781	30,877,456	30,648,675	205,549
	Cavity	17,654	9,755	5,753	31,609,946	31,604,193	7,899
	Crack	279,658	169,594	19,302	31,347,942	31,328,640	110,064
	Blade trail	365,018	192,894	77,039	31,262,582	31,185,543	172,124
wCE ICF	Agglomeration	750,144	686,473	1,334,680	30,877,456	29,542,776	63,671
	Cavity	17,654	15,657	222,499	31,609,946	31,387,447	1,997
	Crack	279,658	236,633	22,574	31,347,942	31,115,368	43,025
	Blade trail	365,018	321,653	1,424,710	31,262,582	29,837,872	43,365
wCE ISCF	Agglomeration	750,144	615,916	394,615	30,877,456	30,482,841	134,228
	Cavity	17,654	13,336	55,297	31,609,946	31,554,649	4,318
	Crack	279,658	225,323	173,822	31,347,942	31,174,120	54,335
	Blade trail	365,018	287,160	617,291	31,262,582	30,645,291	77,858
Focal Loss	Agglomeration	750,144	600,207	388,691	30,877,456	30,488,765	149,937
	Cavity	17,654	9,431	21,615	31,609,946	31,588,331	8,223
	Crack	279,658	195,683	41,485	31,347,942	31,306,457	83,975
	Blade trail	365,018	257,955	273,218	31,262,582	30,989,364	107,063
FL ICF	Agglomeration	750,144	694,448	1,724,581	30,877,456	29,152,875	55,696
	Cavity	17,654	15,604	413,391	31,609,946	31,196,555	2,050
	Crack	279,658	234,522	371,246	31,347,942	30,976,696	45,136
	Blade trail	365,018	320,349	1,457,075	31,262,582	29,805,507	44,669
FL ISCF	Agglomeration	750,144	616,962	408,781	30,877,456	30,468,675	133,182
	Cavity	17,654	13,488	47,525	31,609,946	31,562,421	4,166
	Crack	279,658	226,332	93,258	31,347,942	31,254,684	53,326
	Blade trail	365,018	267,674	311,554	31,262,582	30,951,028	97,344
Binary FL	Agglomeration	750,144	620,326	650,494	30,877,456	30,226,962	129,818
	Cavity	17,654	9,298	15,017	31,609,946	31,594,929	8,356
	Crack	279,658	212,708	67,261	31,347,942	31,280,681	66,950
	Blade trail	365,018	215,811	189,150	31,262,582	31,073,432	149,207
Bin FL ISCF	Agglomeration	750,144	644,750	721,941	30,877,456	30,155,515	105,394
	Cavity	17,654	13,577	72,741	31,609,946	31,537,205	4,077
	Crack	279,658	231,323	105,129	31,347,942	31,242,813	48,335
	Blade trail	365,018	262,401	487,286	31,262,582	30,775,296	102,617

Table A1: Basic metrics results.

Model	Defect	TPR	FPR	Precision	Mean TPR	Mean FPR	Mean Precision
Baseline	Agglomeration	0.6475	0.0072	0.7055	0.5341	0.0036	0.7372
	Cavity	0.3656	0.0002	0.5605			
	Crack	0.6026	0.0011	0.8912			
	Blade trail	0.5208	0.0057	0.7917			
Eq. Samples	Agglomeration	0.6893	0.0064	0.7334	0.6789	0.0055	0.7061
	Cavity	0.5753	0.0003	0.545			
	Crack	0.8237	0.0048	0.8584			
	Blade trail	0.6275	0.0104	0.6876			
Os. Cavities	Agglomeration	0.7326	0.0068	0.7398	0.7196	0.0042	0.7555
	Cavity	0.6954	0.0003	0.6218			
	Crack	0.7254	0.0023	0.8719			
	Blade trail	0.7249	0.0075	0.7883			
Eq. Pixels	Agglomeration	0.2604	0.0007	0.9003	0.2958	**0.0015**	0.7414
	Cavity	0.4285	0.0006	0.4013			
	Crack	0.4078	0.0044	0.7473			
	Blade trail	0.0836	0.0002	0.9167			
Binary	Agglomeration	0.6589	0.0054	0.7601	0.5622	0.0026	0.7333
	Cavity	0.3591	0.0002	0.5657			
	Crack	0.7742	0.0016	0.8419			
	Blade trail	0.4565	0.0031	0.7656			
Bin. Balanced	Agglomeration	0.726	0.0074	0.7368	0.6034	0.0027	**0.7891**
	Cavity	0.5526	0.0002	0.6834			
	Crack	0.6064	0.0006	0.8989			
	Blade trail	0.5285	0.0025	0.8374			
wCE ICF	Agglomeration	0.9151	0.0432	0.3566	**0.8824**	0.0258	0.3389
	Cavity	0.8869	0.007	0.0908			
	Crack	0.8462	0.0074	0.5776			
	Blade trail	0.8812	0.0456	0.3309			
wCE ISCF	Agglomeration	0.8211	0.0128	0.6149	0.7922	0.0099	0.4969
	Cavity	0.7554	0.0017	0.2298			
	Crack	0.8057	0.0055	0.6498			
	Blade trail	0.7867	0.0197	0.4934			
Focal Loss	Agglomeration	0.8001	0.0126	0.6249	0.6852	0.0058	0.6196
	Cavity	0.5342	0.0007	0.3391			
	Crack	0.6997	0.0013	0.8637			
	Blade trail	0.7067	0.0087	0.6505			
FL ICF	Agglomeration	0.9258	0.0559	0.3053	0.8815	0.0319	0.3078
	Cavity	0.8839	0.0131	0.0499			
	Crack	0.8386	0.0118	0.5366			
	Blade trail	0.8776	0.0466	0.3395			
FL ISCF	Agglomeration	0.8225	0.0132	0.6093	0.7823	0.0069	0.5623
	Cavity	0.764	0.0015	0.2619			
	Crack	0.8093	0.003	0.7508			
	Blade trail	0.7333	0.01	0.6272			
Binary FL	Agglomeration	0.8269	0.0211	0.5254	0.6764	0.0075	0.6157
	Cavity	0.5267	0.0005	0.4247			
	Crack	0.7606	0.0021	0.8048			
	Blade trail	0.5912	0.0061	0.7078			
Bin FL ISCF	Agglomeration	0.8595	0.0234	0.5035	0.7937	0.0112	0.4901
	Cavity	0.7691	0.0023	0.1957			
	Crack	0.8272	0.0034	0.7104			
	Blade trail	0.7189	0.0156	0.5508			

Table A2: Metrics results.

Appendix B

Computational costs

Section	Layer	Kernels	Kernel H/W	Stride	Padding	Activations H/W	Activations Channels	#Activations	#Params	Flops (M)	Activations Memory (MB	Skip Connections (MB)	Parameters Memory (MB)
Encoder	input					256	3	196608	0		12		0,00
	conv1	32	3	1	1	256	32	2097152	896	108	256		0,01
	conv2	32	3	1	1	256	32	2097152	9248	1152	256	256	0,14
	pool1	1	2	2	0	128	32	524288	0	2	64		0,00
	conv3	64	3	1	1	128	64	1048576	18496	576	128		0,28
	conv4	64	3	1	1	128	64	1048576	36928	1152	128	128	0,56
	pool2	1	2	2	0	64	64	262144	0	1	32		0,00
	conv5	128	3	1	1	64	128	524288	73856	576	64		1,13
	conv6	128	3	1	1	64	128	524288	147584	1152	64	64	2,25
	pool3	1	2	2	0	32	128	131072	0	0,5	16		0,00
	conv7	256	3	1	1	32	256	262144	295168	576	32		4,50
	conv8	256	3	1	1	32	256	262144	590080	1152	32	32	9,00
	pool4	1	2	2	0	16	256	65536	0	0,25	8		0,00
	conv9	512	3	1	1	16	512	131072	1180160	576	16		18,01
Decoder	conv10	512	3	1	1	16	512	131072	2359808	1152	16		36,01
	up1	1	2	2	0	32	256	262144	0	1	32		0,00
	conv11	256	3	1	1	32	256	262144	590080	1152	32		9,00
	conv12	256	3	1	1	32	256	262144	590080	1152	32		9,00
	up2	1	2	2	0	64	128	524288	0	2	64		0,00
	conv13	128	3	1	1	64	128	524288	147584	1152	64		2,25
	conv14	128	3	1	1	64	128	524288	147584	1152	64		2,25
	up3	1	2	2	0	128	64	1048576	0	4	128		0,00
	conv15	64	3	1	1	128	64	1048576	36928	1152	128		0,56
	conv16	64	3	1	1	128	64	1048576	36928	1152	128		0,56
	up4	1	2	2	0	256	32	2097152	0	8	256		0,00
	conv17	32	3	1	1	256	32	2097152	9248	1152	256		0,14
	conv18	32	3	1	1	256	32	2097152	9248	1152	256		0,14
	out	5	3	1	1	256	5	327680	1445	180	40		0,02
						Minibatch:	16	**Totals:**	6281349	17586,75	2604	480	95,85

Table B1: Baseline model's computational costs. The pink cells highlight the skip connections.

Section	Layer	Kernels	Kernel H/W	Stride	Padding	Activations H/W	Activations Channels	#Activations	#Params	Flops (M)	Activations Memory (MB)	Skip Connections (MB)	Parameters Memory (MB)
Encoder	input					256	3	196608	0		12		0,00
	conv1	32	3	1	1	256	32	2097152	896	108	256		0,01
	conv2	32	3	1	1	256	32	2097152	9248	1152	256	256	0,14
	pool1	1	2	2	0	128	32	524288	0	2	64		0,00
	conv3	64	3	1	1	128	64	1048576	18496	576	128		0,28
	conv4	64	3	1	1	128	64	1048576	36928	1152	128	128	0,56
	pool2	1	2	2	0	64	64	262144	0	1	32		0,00
	conv5	128	3	1	1	64	128	524288	73856	576	64		1,13
	conv6	128	3	1	1	64	128	524288	147584	1152	64	64	2,25
	pool3	1	2	2	0	32	128	131072	0	0,5	16		0,00
	conv7	256	3	1	1	32	256	262144	295168	576	32		4,50
	conv8	256	3	1	1	32	256	262144	590080	1152	32	32	9,00
	pool4	1	2	2	0	16	256	65536	0	0,25	8		0,00
	conv9	512	3	1	1	16	512	131072	1180160	576	16		18,01
Decoder	conv10	512	3	1	1	16	512	131072	2359808	1152	16		36,01
	up1	1	2	2	0	32	256	262144	0	1	32		0,00
	conv11	256	3	1	1	32	256	262144	590080	1152	32		9,00
	conv12	256	3	1	1	32	256	262144	590080	1152	32		9,00
	up2	1	2	2	0	64	128	524288	0	2	64		0,00
	conv13	128	3	1	1	64	128	524288	147584	1152	64		2,25
	conv14	128	3	1	1	64	128	524288	147584	1152	64		2,25
	up3	1	2	2	0	128	64	1048576	0	4	128		0,00
	conv15	64	3	1	1	128	64	1048576	36928	1152	128		0,56
	conv16	64	3	1	1	128	64	1048576	36928	1152	128		0,56
	up4	1	2	2	0	256	32	2097152	0	8	256		0,00
	conv17	32	3	1	1	256	32	2097152	9248	1152	256		0,14
	conv18	32	3	1	1	256	32	2097152	9248	1152	256		0,14
	out	**2**	3	1	1	256	**2**	131072	578	72	16		0,01
						Minibatch:	16	**Totals:**	6280482	17478,75	2580	480	95,83

Table B2: Binary model's computational costs. The pink cells highlight the skip connections.

Bibliography

[1] Doris Antensteiner. and Svorad Štolc. Regularization in higher-order photometric stereo inspection for non-lambertian reflections. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 4: VISAPP*, pages 253–259. INSTICC, SciTePress, 2020.

[2] Arpit Bansal, Micah Goldblum, Valeriia Cherepanova, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Metabalance: High-performance neural networks for class-imbalanced data. *arXiv preprint arXiv:2106.09643*, 2021.

[3] J. Brownlee. *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Machine Learning Mastery, 2020.

[4] Jie Chang, Xiaoci Zhang, Minquan Ye, Daobin Huang, Peipei Wang, and Chuanwen Yao. Brain tumor segmentation based on 3d unet with multi-class focal loss. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5. IEEE, 2018.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[6] Gabriela Csurka, Riccardo Volpi, and Boris Chidlovskii. Semantic image segmentation: Two decades of research, 2023.

[7] Kento Doi and Akira Iwasaki. The effect of focal loss in semantic segmentation of high resolution aerial image. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 6919–6922. IEEE, 2018.

[8] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee, 2008.

[9] Xin He, Yong Zhou, Jiaqi Zhao, Di Zhang, Rui Yao, and Yong Xue. Swin transformer embedding unet for remote sensing image semantic segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022.

[10] Christian Kapeller, Bernhard Blaschitz, and Ernst Bodenstorfer. Inline battery foil inspection using strobed photometric stereo. In *Forum Bildverarbeitung 2020*, United States, 2020. Society for Imaging Science and Technology. Forum Bildverarbeitung 2020 ; Conference date: 26-11-2020 Through 27-11-2020.

[11] Christian Kapeller and Ernst Bodenstorfer. Photometric stereo-based high-speed inline battery electrode inspection. *tm - Technisches Messen*, 88(7-8):423–432, 2021.

[12] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[14] Kangcheng Liu. Semi-supervised confidence-level-based contrastive discrimination for class-imbalanced semantic segmentation. In *2022 12th International conference on CYBER technology in automation, control, and intelligent systems (CYBER)*, pages 1230–1235. IEEE, 2022.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] Ningthoujam Johny Singh and Kishorjit Nongmeikapam. Semantic segmentation of satellite images using deep-unet. *Arabian Journal for Science and Engineering*, 48(2):1193–1205, 2023.

[19] Ruoqi Sun, Xinge Zhu, Chongruo Wu, Chen Huang, Jianping Shi, and Lizhuang Ma. Not all areas are equal: Transfer learning for semantic segmentation via hierarchical region selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4360–4369, 2019.

[20] The GIMP Development Team. Gimp.

[21] Libo Wang, Rui Li, Ce Zhang, Shenghui Fang, Chenxi Duan, Xiaoliang Meng, and Peter M Atkinson. Unetformer: A unet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 190:196–214, 2022.

[22] Robert Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19, 01 1992.

[23] Binhui Xie, Longhui Yuan, Shuang Li, Chi Harold Liu, and Xinjing Cheng. Towards fewer annotations: Active learning via region impurity and prediction uncertainty for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8068–8078, 2022.

[24] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31, 2018.