# Machine Learning in Applications
# Tool wear classification
# Project Report

Claudio Tancredi (292523), Francesca Russo (287935), Alessandro Versace (292435), Matteo Quarta (292477)
Politecnico di Torino

CONTENTS

LIST OF FIGURES

## List of Tables

# Machine Learning in Applications
# Tool wear classification
# Project Report

***Abstract*—Monitoring tools is a fundamental process in modern factories because, by substituting worn tools with new ones, economic losses can be prevented. By using well known architectures able to solve both instance segmentation and classification tasks at the same time, it is possible to automate such process. However, manual labeling of the actual state of each worn region is time-consuming and requires the presence of highly qualified human operators. In this paper, an existing model, called baseline, has been implemented and its results have been analyzed. Then, an attempt to lighten the labeling phase is proposed, by building a two-stage pipeline architecture where the first stage focuses on wear localization and extraction by means of an architecture for instance segmentation, while the second stage performs the actual state classification for tools. In addition, to evaluate the feasibility of a real-time monitoring application, two different models that run at different fps were tested as stage 1 models. Results show that the proposed two-stage pipeline significantly outperforms the analyzed baseline model and that real-time applications are viable whenever speed is preferred over prediction correctness.**

## I. Introduction

Nowadays manufacturing industries use cemented carbide inserts as cutting tools for the final steps of the production line, when refinements are needed for the produced objects. However, these inserts tend to get damaged or wear out due to intensive use, leading to low quality or defected products. Supervision of worn out inserts thus becomes a crucial task, allowing to replace the worn cutting insert with a new one, making it possible to keep high production quality standards and yield and, at the same time, to greatly reduce costs. This task is usually carried out by highly qualified human operators, exposing the monitoring process to several risks: human errors, low operator expertise, subjective decision process, etc. Machine Learning and Computer Vision can significantly improve this process, by automatically detecting worn out areas and giving estimates about the state of an insert, deciding whether it should be replaced or not.

In this work we are going to evaluate different approaches to automate the above-mentioned monitoring process.

All the presented models have been trained and tested on a new dataset, which includes images of four different types of cemented carbide inserts:

- RNGN19
- RNGN12
- CNGA12
- RCGX12

listed in order of increasing number of samples at our disposal.

Images were provided with their rated state, either ”Ok”, ”Not Ok” or ”Doubt”, which from now on we will be referring to as OK, NOK and DOUBT, but without any information regarding the classification process and with no bounding boxes and masks annotations whatsoever. Therefore, COCO [1] format annotations for both masks and bounding boxes were manually generated by us. Certainly, existing and well known architectures for instance segmentation, such as Mask R-CNN [2], seem suitable and promising options to solve the problem. Nonetheless, some aspects need to be considered.

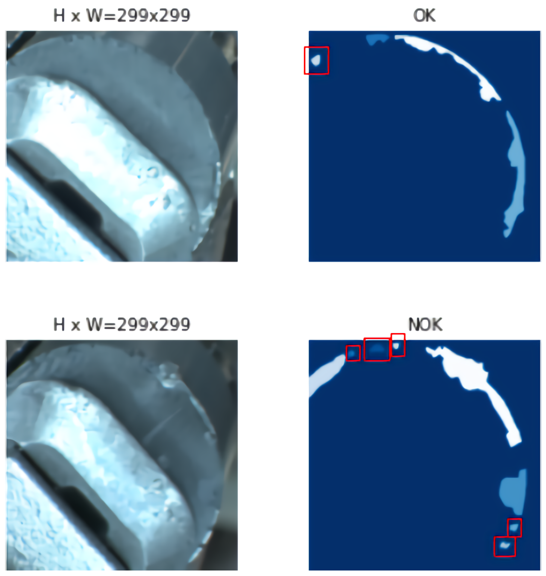

Fig. 1: Left: component image. Right: corresponding ground truth instance segmentation masks and labels. The masks inside the red boxes have different labels (top: OK; bottom: NOK) but show similarities in terms of shape and dimension. It is clear that the worn regions of the NOK image, if taken singularly and classified by a highly qualified human operator, would have probably been annotated as OK.

First of all, attention should be given to architecture performances at inference time, making sure that machinery analysis is performed as quickly as possible to guarantee that it does not become a bottleneck in the production pipeline or requires high operative costs. For this reason, in addition to Mask R-CNN we will also test YOLACT [3], a fully-convolutional model for real-time instance segmentation, to evaluate a possible compromise between correctness and speed.

The evaluation of the state of an insert is, for a qualified operator, a relatively fast task. The same cannot be said for the segmentation annotation process, which is a rather long process. The presence of a highly qualified operator is crucial for correctly labeling worn regions since, within the same image, regions with different levels of wear (OK, NOK, DOUBT) may be present, forcing manufacturing industries to face large expenses for an extended time to compensate for operators' valuable skills. For these reasons, in our work we are going to investigate a possible solution to these problems, by allowing the labeling of worn regions to be performed by operators with little or no domain skills at all and by simplifying the task itself, so that it can be completed as soon as possible.

As a consequence of the lack of instance segmentation annotations, and given that we claim no expertise in this specific field to properly annotate worn regions as OK, NOK or DOUBT, the labeling phase turned out to be critical. By solely relying on the information about the actual state of a component, all regions that seemed worn out were labeled according to the claimed state. This may have resulted in inconsistent mask annotations for the dataset, because small or medium-sized regions can be labeled as OK, NOK or DOUBT depending on the image they appear in. This can be seen in Figure 1.

To mitigate this issue, which may greatly affect models reliability, we are going to emulate the cognitive process of a highly qualified human operator that, in most of the cases, consists of determining the state of a component based on the largest worn region. To do so, we propose a two-stage pipeline where the first stage performs instance segmentation and whose objective is to identify the worn regions; then, the largest region is passed down to the second stage, that includes a Convolutional Neural Network (CNN) for image classification that can now focus only on the most relevant region of each image and provide final predictions about component states. In this way, we both diminish the need for highly qualified operators in the labeling phase and solve the problem of having inconsistent mask labels for the dataset.

The paper is organised in the following sections: Background, which gives an overview of the current domain context; Materials and methods, that extensively illustrate our research; Results and discussion, in which our work is analyzed in detail; Conclusions and future works, which serves the purpose of summarising the key points of our work and pave the way for possible improvements.
Code is available at:

https://github.com/claudiotancredi/Machine-learning-in-applications.

## II. Background

In a world of ever-increasing computational power, the existence of the Smart Factory concept within the so-called Industry 4.0 revolution should not come as a surprise [4].

This concept lays its foundation in the idea that in order to cope with an ever growing demand for goods, both in quantity and variety, production costs must be reduced as much as possible. This requires maximizing yield, by reducing the number of finished products the manufacturers won't be able to sell and, at the same time, by reducing production downtime, which by itself has some costs.

Studies have been conducted to monitor production and detect anomalies, in which patterns are analyzed in order to predict and prevent critical situations that may occur, by examining physical quantities such as heat [5]. Another approach investigated in previous studies is based on acceleration vibration signals, which have been found significant to identify tool wear condition during the turning of gray cast-iron EN-GJL-250 using carbide cutting inserts [6]. On a different path, [7] classifies tool inserts wear level using a vision system, even though the process of classification is based on shallow Machine Learning techniques. Nonetheless, it is interesting to notice the pre-processing applied on the acquired images to highlight the worn region: background blurring, contrast enhancement, segmentation, noise reduction, etc. While there are some notable differences, the idea of focusing on the worn region is somewhat shared with our two-stage pipeline approach.

## III. Materials and methods

To carry out our research we have identified four main steps: dataset analysis, dataset labeling, implementation of both a baseline model and a two-stage pipeline, generalization tests. These are explained in detail in the following subsections.

### A. *Step 1 - Dataset analysis*

Dataset images have been acquired with a special camera that allowed to collect 9 different images in a 3x3 grid format using a single robot pose, making it possible to capture several perspectives for the same insert. Then, these images have been examined to assess their quality and corner images in the 3x3 grid have been discarded as not completely being in focus, keeping only 5 of the original 9 images. This acquisition process gives us the opportunity to evaluate the state of the insert by combining the information of its 5 images given by the camera, but a more conservative approach has been considered here, which may turn out to be a better choice.

Indeed, combining data information would require industries to have always access to this special equipment and it may not always be the case, thus making the architecture less generic. For these reasons, in our work, images belonging to the same set of photos taken from a single insert at a given time have been considered as independent both at training and inference phases, as if they were different photos acquired on different inserts at different times. This also implies that two images, belonging to the same set of 5 photos taken on a carbide insert, could be labeled differently, but we do not believe this to be a problem.

On the contrary, this represents by itself some sort of beforehand data augmentation and, on the other hand, it allows models to succeed also in the worst-case scenario of a very simple equipment for data acquisition. Then, if industries do have access to special equipment, it would also be really easy to add "final call" decision mechanisms, such as majority voting.

Another aspect to take in consideration is the distribution of classes in the dataset. Figure 2 shows the class distributions of the four insert types that compose the dataset. It is evident that, in most cases, the dataset is highly unbalanced, with OK as the dominant class. This is a domain specific problem: if we consider the lifetime of a high quality insert, most of the time is spent in the OK state, while less time is spent in DOUBT or NOK states, till the replacement. This makes the acquisition of OK images more frequent with respect to DOUBT or NOK images. This may lead, as we will see further on, to models that voluntarily ignore rare classes. Therefore, for model analysis, at training time we will use both the original and a rebalanced dataset, while tests will be conducted on an unbalanced set because that is the model application context.

## B. Metrics

A variety of standard metrics available in literature have been used to evaluate the goodness of our work:

- Mask mAP @ 0.50 (%)
- Accuracy (%), $\frac{\text{Correctly classified samples}}{\text{Total number of samples}}$
- Per class accuracy (%), defined as $\frac{\text{Correctly classified samples for class c}}{\text{Total number of samples of class c}}$.

Since the dataset is unbalanced, accuracy has no real meaning for the task and may probably be unreliable. Other metrics, such as balanced accuracy, might be preferred, but they would still focus on an overall view and give no indication about model performance for a single class. Given that for industries the most relevant use case is to identify NOK components due to the costs that wrong predictions on inserts that need to be replaced might imply, there is clearly a need to distinguish per class model performance. For this reason, the main reference metric will be per class accuracy.

In the light of the importance of the NOK use case, in addition to these standard metrics we also introduce two new custom metrics for the task:

- Critical predictions (also called Crit for convenience) - percentage (%) of NOK images classified as OK;

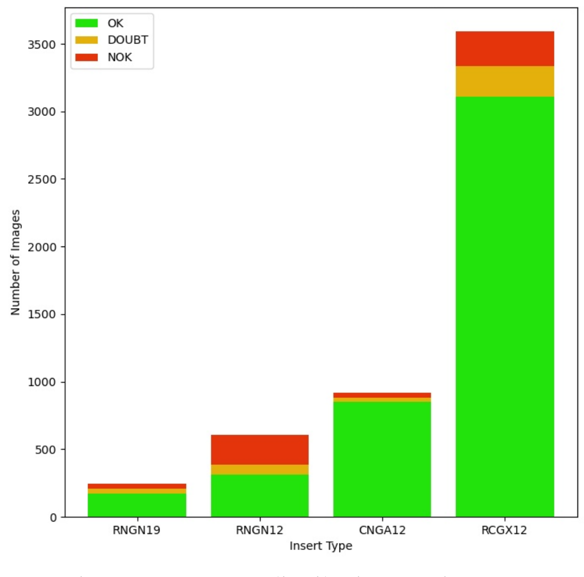

Fig. 2: Wear state distribution per insert type

- Semi-Critical predictions (also called Semi-Crit for convenience) - percentage (%) of NOK images classified as DOUBT.

The first metric is used to measure a severe model failure, since a *Critical* classification may lead to unscheduled maintenance and low quality products, or result in a significant loss in production yield. The second one, *Semi-Critical*, is more related to the factory management, and so whether a DOUBT classification is checked immediately or scheduled to be checked in a certain time interval by a highly qualified human operator, or even not checked at all.

## C. Step 2 - Dataset labeling

As already mentioned, dataset labeling followed two separate processes, the first one conducted by qualified personnel verifying the state of the insert, and a second one which required us to manually annotate worn regions and generate instance segmentation annotations. Each generated mask is associated to a state, either OK, NOK or DOUBT, and the annotations chosen format is COCO [1].

An implicit problem with this method is the absence of feedback in mask annotations. It is not uncommon for a worn out insert to have multiple degraded areas, and the decision-making process in identifying and marking them can be considered objective to some extent. What is not objective, however, is whether all worn areas have to be labeled the same for an image with multiple wear instances, or if different labels (OK, DOUBT, NOK) can be assigned to these instances. As already pointed out in the introduction, this heterogeneous mask labeling process is not possible for non-qualified operators. By solely relying on the information about the actual state of a component, all regions that seemed worn out were labeled according to the claimed state. This may have resulted in inconsistent mask annotations and, to evaluate the possible influence of this labeling approach, we will test a baseline model, Mask R-CNN [2]. Then, annotations will undergo a translation phase where OK, DOUBT and NOK classes will all be mapped to a single and more generic WEAR class. This logical step is the starting point of our work and makes it possible to remove any inconsistencies and diminish the need for highly qualified human operators for the instance segmentation labeling step, as the wear recognition and labeling process is easy and almost objective.

Specifically, it is now possible to follow a ”divide and conquer” approach to solve the proposed task by dividing it into two simpler sub-tasks: the first one focuses on wear localization, while the second one moves on to wear classification. The resulting advantages are many: ease of understanding, less time-demanding labeling process, inconsistencies resolution in the labeling phase and, consequently, also the removal of the need for highly qualified operators.

## D. Instance segmentation architectures

*1) Mask R-CNN:* One of the two employed architectures for instance segmentation is Mask R-CNN, which consists of two stages. The first stage, called Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage extracts features using RoiAlign, that faithfully preserves spatial locations from each candidate box and performs classification and bounding box regression. In parallel, Mask R-CNN also outputs a binary mask for each RoI.

The convolutional backbone architecture used for feature extraction over an entire image is ResNet-101 [8].

Model training relies on a multi-task loss on each sampled RoI defined as: $L = L_{cls} + L_{box} + L_{mask}$, in which $L_{mask}$ is an average binary cross-entropy loss, $L_{box}$ is a smooth L1-loss and $L_{cls}$ is the classification loss defined as a cross-entropy loss.

The number of trainable parameters for Mask R-CNN [2] is about 63 millions and it runs at 5 fps. Appendix A shows the most relevant configuration parameters for Mask R-CNN.

Mask R-CNN, whenever used in our work, has been initialized with COCO [1] weights and the training phase has been conducted in two steps: first the head layers, so that the model can learn to produce the desired predictions; then fine-tuning on all the layers to refine the model as a whole. This transfer learning approach is usually useful for datasets that share obvious similarities, where a training step may benefit from knowledge extracted from a previous training on a similar dataset. Although our dataset has a low compatibility with COCO, it is also short in the number of images and this makes us believe that model performance may benefit from this initialization and transfer learning technique.

*2) YOLACT:* Mask R-CNN requires re-pooling features for each RoI and process them with subsequent computations, which makes it unable to obtain real-time speed. YOLACT [3] forgoes an explicit localization step and accomplish the instance segmentation task by breaking it into two parallel tasks: (1) generation of a set of prototypes and (2) prediction of the per-instance mask coefficients. Then, the instance masks are produced by linearly combining the prototypes with the mask coefficients. This process allows YOLACT to reach at least 30 fps.

The backbone architecture used for feature extraction over an entire image is ResNet-101. The number of trainable parameters for YOLACT [3] is about 42 millions. In our work, pre-trained YOLACT weights achieving $\sim$ 40mAP on COCO are used as starting points.

The model is trained by using three losses: classification loss $L_{cls}$, box regression loss $L_{box}$ and mask loss $L_{mask}$. The classification loss is a softmax loss, the box regression loss is a smooth L1-loss and the mask loss is the pixel-wise binary cross-entropy.

### *E. Classification architecture - ResNet-50*

A simple ResNet-50 [8] classification network has been chosen, with pre-trained ImageNet weights. In the classification step with ResNet-50, annotations with OK, DOUBT and NOK classes are restored and used.

Training is carried out by minimizing a simple multi-class crossentropy loss and follows the same transfer learning approach described in III-D1 for the same reasons.

### *F. Step 3 - Baseline and two-stage pipeline implementations*

The train-validation-test split chosen for both the baseline model and the pipeline architecture is:

- 70% training set
- 10% validation set
- 20% test set.

For the sake of comparison, the sets of training-validation-test images used for both the baseline model and the pipeline architecture are exactly the same.

As for the unbalanced dataset, whenever 3-classes annotations (OK, DOUBT, NOK) are used, a rebalanced version of the training set will be employed in addition to the unbalanced one. Rebalancing is performed by cutting off excess OK images according to:

$$\text{\# of OK samples} = \frac{\text{\# of NOK samples + \# of DOUBT samples}}{2}$$

The resulting rebalanced datasets may be, in some cases, too small. This may be a problem and lead to overfitting. To compensate for the limited number of training samples that models may deal with, image augmentation has been used. Specifically, horizontal and vertical flips, and rotation of 90°, have been used not only when dealing with rebalanced training sets, but also with unbalanced training sets, to further generalize for the task.

Since our goal is to emulate the cognitive process of a qualified human operator, both the baseline model and the two-stage pipeline will follow the same procedure at inference time: identification of the worn region that most likely captures the operator's attention and drives his/her decision, which usually corresponds to the largest predicted segmentation mask, followed by state classification.

*1) Baseline model - Mask R-CNN:* A baseline Mask R-CNN model is the most naive approach that can be used to solve the proposed task. Having the wear state labels and instance segmentation annotations, one could think to just run an instance segmentation model and that it should suffice. We're going to test this hypothesis and show evidence that it is not the optimal strategy.

Appendix B presents the most relevant configuration training parameters for Mask R-CNN baseline model.

Model weights are stored during training. Precisely, best epoch weights saving strategy has been implemented. Moreover, early stopping with a patience parameter equal to 20 has been used, to stop the training at the very first signs of overfitting.

Several instance segmentation predictions may be produced at inference time and a final decision process is needed to estimate the insert state. As anticipated, the largest predicted segmentation mask is selected and its predicted class is what eventually determines the insert state. Appendix C collects the most relevant configuration inference parameters for Mask R-CNN baseline model.

*2) Two-stage models:* The two-stage models implement a rather simple pipeline, as it can be seen in Figure 3, in which the first stage is represented by either Mask R-CNN

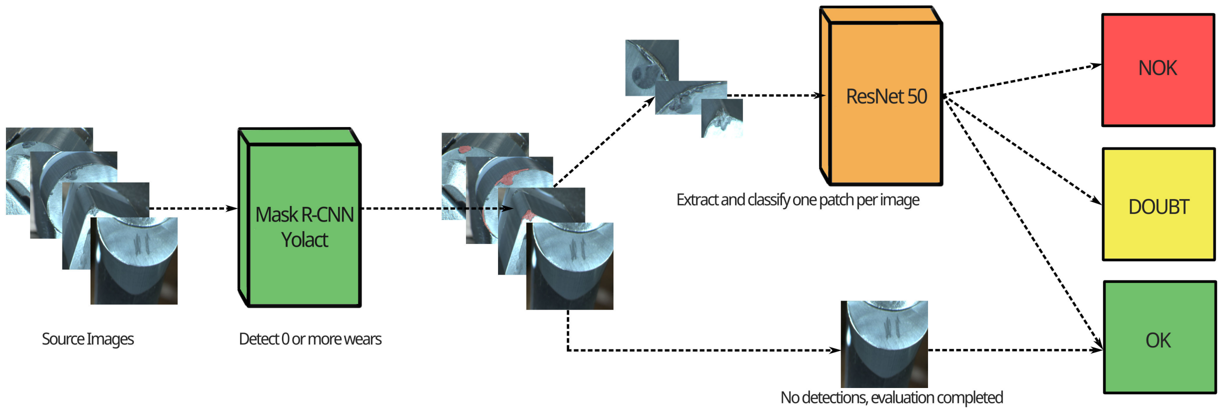
Source Images
Mask R-CNN
Yolact
Detect 0 or more wears
Extract and classify one patch per image
ResNet 50
NOK
DOUBT
OK
No detections, evaluation completed

Fig. 3: Pipeline Architecture. The figure shows how predictions are made by the pipeline. When no worn regions are detected in the first stage, the insert is automatically classified as OK. When worn regions are detected, the bounding box related to the segmentation mask with the largest area is used to crop the image and pass the crop down to stage 2, which uses ResNet-50 to perform image classification.

or YOLACT and can be summarized as a wear detection network, while the second stage classifies the state of said wear regions. Given that the first stage works only with 2 classes, BACKGROUND and WEAR, rebalancing is not needed.

The first stage behaves differently and produces different outputs at training and test time for the pipeline.

While training the pipeline, for each training and validation image, the first stage detects as much worn regions with confidence greater than 0.8 as it can, but passes only one image crop to the second stage. The criterion used for choosing such crop is based on both the confidence of the prediction and the predicted masks areas. Among all selected masks that have a confidence over a chosen threshold, set to 0.95 in our work, the bounding box of the one with the biggest area is used to crop the image and the crop is passed down to stage 2. It is interesting to notice that predictions are not usually performed over training and validation sets, but in our case study it is necessary to do so to make the most of the images at our disposal. Cropped images will then serve as a training set for the second stage, which will create and work with its own split. Moreover, by selecting a high threshold we make sure that only really confident predictions are passed down to the second stage, which can rely on high quality data. In fact, images for which no reliable masks are detected are, in practice, discarded and not considered by stage 2. By doing so, the number of samples slightly decreases, but it's a cost we're willing to pay for the stage 2 to properly operate.

Appendix D shows the most relevant configuration training parameters for Mask R-CNN used as stage 1 model.

For YOLACT, training is conducted for a variable number of epochs between 150 and 300, depending on the insert being tested (bigger dataset, lower epochs). Training is conducted up till the end of all epochs, best model with respect to a mAP @ 0.50 metric computed on a validation split is stored and used for stage 2. No further data augmentation rather than the default provided one by the YOLACT GitHub [9]. No momentum or weight decay is used. Other relevant parameters for YOLACT as stage 1 model can be found in appendix F.

At inference time the process is similar. For each test image the first stage detects as much worn regions with confidence greater than 0.8 as it can, but passes only one image crop to the second stage. The criterion used for choosing such crop is based only on the predicted masks areas. Among all the detected masks, the bounding box of the one with the biggest area is used to crop the image and the crop is passed down to stage 2 for classification. In addition to cropped images, a JSON file is also generated and made accessible to stage 2 with useful information for final predictions visualization. Appendix E shows the most relevant configuration inference parameters for Mask R-CNN used as stage 1 model.

For the second stage, cropped images vary in size, perhaps even by an order of magnitude, and need to be resized to a fixed shape, which for us is 120×120. Dropout is applied as a regularization method right after global average pooling.

## G. *Step 4 - Generalization tests*

Doubts regarding the generalization capabilities of this automated supervision process may arise when considering the variety of inserts that may be used in a production line.

For this reason, we also investigated the capability of our two-stage pipeline to correctly predict the state of an insert different than the one it was initially trained on, without any kind of extra training steps. Moreover, we also evaluated results after a fine-tuning step on the new component to recreate the scenario in which images of a new insert may not be available in great quantities.

The fine-tuning step of the pipeline is peculiar: first, the stage 1 model pre-trained on an insert undergoes the training process for head layers on the new insert; then, as already discussed, predictions are made on training and validation sets, and cropped images are passed to stage 2; finally, the stage 2 model pre-trained on exactly the same insert sees its top layers trained on the new insert. Only then, evaluation can be performed.

The idea is to examine both a best-case and a worst-case scenario, so the tests that we will conduct are:

- evaluation of RNGN12-trained pipeline (cylindrical insert) on RNGN19 (cylindrical insert); fine-tuning

TABLE I: Baseline model - Mask R-CNN Results

All values are percentages.

| Insert Type | Rebalanced Dataset | | | | Unbalanced Dataset | | | |
|---|---|---|---|---|---|---|---|---|
| | Wear Level | Accuracy | Semi-Crit | Crit | Wear Level | Accuracy | Semi-Crit | Crit |
| RNGN19 | OK | 94.44 | 0.00 | **66.67** | OK | 94.44 | 0.00 | 83.33 |
| | NOK | **33.33** | | | NOK | 16.67 | | |
| | DOUBT | 20.00 | | | DOUBT | 0.00 | | |
| | Overall | 78.23 | | | Overall | 74.47 | | |
| RNGN12 | OK | 91.49 | 0.00 | **17.02** | OK | 100.00 | 0.00 | 21.28 |
| | NOK | **82.98** | | | NOK | 78.23 | | |
| | DOUBT | 0.00 | | | DOUBT | 0.00 | | |
| | Overall | 70.69 | | | Overall | 72.41 | | |
| CNGA12 | OK | 35.39 | 0.00 | **33.33** | OK | 100.00 | 0.00 | 100.00 |
| | NOK | **66.67** | | | NOK | 0.00 | | |
| | DOUBT | 0.00 | | | DOUBT | 0.00 | | |
| | Overall | 35.38 | | | Overall | 86.92 | | |
| RCGX12 | OK | 93.22 | 14.28 | **8.92** | OK | 100.00 | **0.00** | 50.00 |
| | NOK | **76.78** | | | NOK | 50.00 | | |
| | DOUBT | 32.00 | | | DOUBT | 2.00 | | |
| | Overall | 87.25 | | | Overall | 88.45 | | |

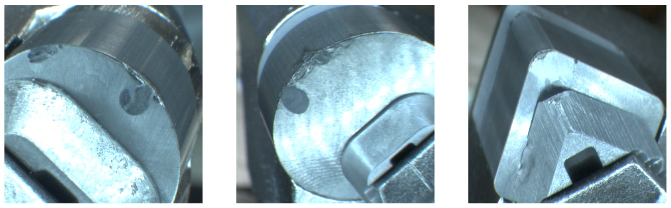
Fig. 4: Inserts used for generalization tests. Left: RNGN12 insert. Center: RNGN19 insert. Right: CNGA12 insert.

of RNGN12-trained pipeline (cylindrical insert) on RNGN19 insert (cylindrical insert), followed by evaluation;

- evaluation of RNGN12-trained pipeline (cylindrical insert) on CNGA12 (parallelepiped-shaped insert); fine-tuning of RNGN12-trained pipeline (cylindrical insert) on CNGA12 (parallelepiped-shaped insert), followed by evaluation;

where the first test belongs to a best-case scenario due to similarities in terms of shape between the two types of insert, while the second one belongs to a worst-case scenario because of the substantial differences in the appearance of the inserts. Similarities and differences can be observed in Figure 4.

## IV. Results and discussion

### A. *Baseline model - Mask R-CNN*

As expected, baseline model performances, reported in table I, are quite poor. This can be easily explained if we consider both the objective function and the training execution environment: considering the minimization of a function for three tasks, boxes, masks and classes, that have unitary weight, and a class distribution as described in Figure 2, always predicting OK is a good way of achieving decent class loss, leaving the function dominated by boxes and masks loss.

For the unbalanced version of the dataset, this leads to an overly confident model biased towards the OK class, with high critical predictions and almost total avoidance of the DOUBT class.

Using a rebalanced version of the dataset and dealing with a more even class distribution, in most cases per class accuracy for NOK and DOUBT classes shows significant improvements and this change is also reflected on our custom metric critical predictions, which drops significantly.

It can be tentatively concluded that dataset rebalancing may improve model performance to some extent, while still being far from impressive or viable at all, especially when working with a limited number of samples (RNGN19) or with inserts that appear to be difficult to analyze (CNGA12).

### B. *Stage I - Mask R-CNN and YOLACT*

Table II shows the results obtained by detecting worn regions on the available inserts. There, some interesting patterns can be noticed.

As the number of training samples increases, performance of Mask R-CNN improves. This may be due to the huge number of learnable parameters of Mask R-CNN and it seems to point to the fact that, when the number of training samples is very limited, the model is unable to generalize well on unseen data.

A notable case is RNGN19, where YOLACT performs much better than Mask R-CNN. This could be related to the dataset size, and being YOLACT a simpler model compared to Mask R-CNN it could do better when available data is scarce.

Apart from the particular case of RNGN19, Mask R-CNN seems to outperform YOLACT. Since stage 2 learning and test results are driven by stage 1 correctness, from the point of view of the final pipeline results Mask R-CNN appears to be the most promising choice for stage 1. However, we do not feel to preclude the possibility to choose YOLACT in the presence of different needs, for example speed over mAP, because its results seem valid enough and it runs much faster than Mask R-CNN, allowing real-time inserts monitoring.

### C. *Two-stage models*

Final classification metrics, for both Mask R-CNN and YOLACT, can be found in table III. Bold format is used to highlight the best results for a given model and a given insert type, between the unbalanced and rebalanced versions

TABLE II: Stage I Instance Segmentation Results

All values are percentages.

| Insert Type | mAP @ 0.50 | |
|---|---|---|
| | Mask R-CNN | Yolact |
| RNGN19 | 47.7 | **62.31** |
| RNGN12 | **59.2** | 49.00 |
| CNGA12 | **78.6** | 39.76 |
| RCGX12 | **78.9** | 78.02 |

of the dataset. Red color points out the best results for a given insert type and a given version (unbalanced/rebalanced) of the dataset, between Mask R-CNN and YOLACT models.

Results seem to be consistent considering that both models receive the same input images: we can clearly see that, for both models, learning how to classify some inserts is harder than others (e.g., CNGA12).

As expected, YOLACT-based pipeline tend to achieve high NOK accuracy and low critical predictions when data is scarce. This is not surprising at all and it is an obvious consequence of stage 1 good performance achieved by YOLACT in wear recognition. In most of the other cases, Mask R-CNN-based pipeline provides more accurate predictions.

We can also notice that the unbalanced dataset leads to the best results. However, it would probably be safer to assume that more in-depth analysis is needed here, since there may be several factors influencing the outcome, such as limited number of training samples, etc.

Finally, we can compare the baseline model I with our two-stage pipeline. It is evident that, given the possibility of having inconsistencies in annotations with classes OK, NOK and DOUBT, our doubts regarding the effectiveness of the baseline model were well founded. In fact, our architecture, which allows us to overcome these problems, significantly outperforms the baseline model, especially for the metrics that are more relevant for the industrial domain, like NOK accuracy, semi-critical and critical predictions. This comes at a cost: it is important to remember that the introduction of a second stage clearly adds an overhead, but the achieved results make it seem fair enough.

Given that Mask R-CNN performance on the unbalanced dataset are the best in our context, generalization tests will be conducted on this specific model and in the same conditions.

### D. Generalization tests

As the results in table IV show, both evaluations (No FT, FT) perform better when the inserts share some common ground (RNGN12, RNGN19). This is most likely due to the kind of wear that two similar inserts may develop, for example two round-shaped inserts can only get damaged along their borders, while square-shaped inserts do it mostly along their corners.

Even if two-stage pipeline results are not thrilling, an interesting observation can be done combining the generalization tests results for the best case scenario with the ones contained in table II. Notice how, after fine-tuning, the network actually achieves a better mAP in stage 1 than the one we got by training only on the desired insert and by starting from COCO weights. This is possible because the two tasks on the two different inserts clearly have similarities and having performed a training step on a different insert makes it possible to benefit from shared learned knowledge.

## V. Conclusions and future works

The labeling process is a critical step for the development of a model able to identify and classify worn regions because of possible inconsistencies in the annotations.

Existing models may not be able to overcome this issue. In particular, Mask R-CNN, which we tested, turned out not to be robust against this ambiguity in annotations and produced insufficient results.

With our two-stage pipeline architecture, we've been able to remove any sources of ambiguity that may possibly derive from the labeling phase, obtaining at the end interesting and better results, especially for NOK class which is the most relevant in the considered industrial context.

The possibility to label worn regions with a generic WEAR class both removes ambiguities in assigning OK, NOK and DOUBT labels, and allows industries to save human and economic resources because the labeling phase can now be carried out by non-qualified personnel.

We tested two different models for our stage 1, analyzing possible compromises between speed and correctness.

The classifier of the second stage is built on top of the architecture of the previous stage. Future works may focus on how to directly integrate the classifier in the stage 1 architecture, but we expect issues to arise and that the new architecture will need adjustments and won't work in the exact same way of our proposed pipeline. Moreover, with our implementation we have decoupled the two stages, allowing to explore different options for stage 1 as long as the stage 2 receives data in a specific format. By integrating the second stage inside an instance segmentation architecture this won't be possible anymore, since each architecture may work in a different way and the second stage may need to be adjusted accordingly.

The generalization tests we conducted are too few to really be able to affirm anything with certainty, so other works may focus on these experiments.

For computational reasons, our rebalancing strategy consists in downsampling the OK class in a very simple way. To better investigate rebalancing advantages, if any, other strategies may be considered, like oversampling DOUBT and NOK classes.

At the end, our solution seems not only suitable for our application, but it can probably be used in other similar situations in the industrial field, where a monitoring process is needed. This reinforces the validity and versatility of our approach.

## References

[1] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017, cite arxiv:1703.06870Comment: Technical report. [Online]. Available: http://arxiv.org/abs/1703.06870

TABLE III: Two-stage models Results

All values are percentages.

| Model | Insert Type | Unbalanced Wear State | Unbalanced Accuracy | Unbalanced Semi-Crit | Unbalanced Crit | Rebalanced Wear State | Rebalanced Accuracy | Rebalanced Semi-Crit | Rebalanced Crit |
|---|---|---|---|---|---|---|---|---|---|
| Mask R-CNN | RNGN19 | OK | 94.44 | 0.00 | 50.00 | OK | 80.34 | 0.00 | 50.00 |
| | | NOK | 50.00 | | | NOK | 50.00 | | |
| | | DOUBT | 100.00 | | | DOUBT | 100.00 | | |
| | | Overall | 89.36 | | | Overall | 80.85 | | |
| | RNGN12 | OK | 89.36 | **2.13** | **2.13** | OK | 95.75 | 6.38 | 8.51 |
| | | NOK | **95.74** | | | NOK | 85.11 | | |
| | | DOUBT | 63.64 | | | DOUBT | 45.45 | | |
| | | Overall | 87.07 | | | Overall | 81.90 | | |
| | CNGA12 | OK | 91.15 | **0.00** | **22.22** | OK | 77.88 | 11.11 | 44.44 |
| | | NOK | **77.78** | | | NOK | 44.44 | | |
| | | DOUBT | 37.50 | | | DOUBT | 62.50 | | |
| | | Overall | 86.92 | | | Overall | 74.62 | | |
| | RCGX12 | OK | 96.97 | **5.36** | **7.14** | OK | 97.68 | 25.00 | 41.07 |
| | | NOK | **87.50** | | | NOK | 33.93 | | |
| | | DOUBT | 76.00 | | | DOUBT | 24.00 | | |
| | | Overall | 94.60 | | | Overall | 86.81 | | |
| Yolact | RNGN19 | OK | 100.00 | 0.00 | 0.00 | OK | 91.67 | 0.00 | 0.00 |
| | | NOK | 100.00 | | | NOK | 100.00 | | |
| | | DOUBT | 100.00 | | | DOUBT | 100.00 | | |
| | | Overall | 100.00 | | | Overall | 93.62 | | |
| | RNGN12 | OK | 100.00 | **2.17** | **8.70** | OK | 95.74 | 6.52 | 13.04 |
| | | NOK | **89.32** | | | NOK | 80.85 | | |
| | | DOUBT | 68.18 | | | DOUBT | 72.73 | | |
| | | Overall | 89.66 | | | Overall | 85.34 | | |
| | CNGA12 | OK | 92.92 | **0.00** | 62.50 | OK | 86.73 | 25.00 | **37.50** |
| | | NOK | 44.44 | | | NOK | 44.44 | | |
| | | DOUBT | 0.00 | | | DOUBT | 50.00 | | |
| | | Overall | 83.85 | | | Overall | 81.54 | | |
| | RCGX12 | OK | 99.47 | 12.00 | 18.00 | OK | 98.75 | **10.00** | 18.00 |
| | | NOK | 73.21 | | | NOK | **75.00** | | |
| | | DOUBT | 26.00 | | | DOUBT | 28.00 | | |
| | | Overall | 91.75 | | | Overall | 91.45 | | |

TABLE IV: Generalization tests Mask R-CNN Two-stage pipeline Results (Unbalanced Dataset)

"No FT" metrics are computed on the target insert with no fine-tuning steps, while "FT" metrics include the fine-tuning process.
All values are percentages.

| Insert Type | Stage I mAP @ 0.50 No FT | Stage I mAP @ 0.50 FT | Wear State | Two-stage pipeline (No FT) Accuracy | Two-stage pipeline (No FT) Semi-Crit | Two-stage pipeline (No FT) Crit | Two-stage pipeline (FT) Accuracy | Two-stage pipeline (FT) Semi-Crit | Two-stage pipeline (FT) Crit |
|---|---|---|---|---|---|---|---|---|---|
| RNGN12 $\rightarrow$ RNGN19 | 38.40 | **54.10** | OK | 52.78 | 0.00 | 33.33 | 91.67 | 0.00 | 33.33 |
| | | | NOK | 66.67 | | | 66.67 | | |
| | | | DOUBT | 20.00 | | | 80.00 | | |
| | | | Overall | 51.06 | | | 87.23 | | |
| RNGN12 $\rightarrow$ CNGA12 | 24.20 | **65.60** | OK | 52.21 | 0.00 | **66.67** | 98.23 | 0.00 | 100.00 |
| | | | NOK | **33.33** | | | 0.00 | | |
| | | | DOUBT | 0.00 | | | 0.00 | | |
| | | | Overall | 47.69 | | | 85.38 | | |

[3] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation." in *ICCV*. IEEE, 2019, pp. 9156–9165. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2019.html#BolyaZXL19

[4] W. MacDougall, "Industrie 4.0: Smart manufacturing for the future," Germany Trade and Invest, Gesellschaft für Außenwirtschaft und Standortmarketing mbH, Brochure 20750, Jul. 2014, reprint May, 2016. [Online]. Available: http://www.gtai.de/GTAI/Navigation/EN/Invest/Service/publications,did=917080.html

[5] M. Abdallah, B.-G. Joung, W. J. Lee, C. Mousoulis, J. W. Sutherland, and S. Bagchi, "Anomaly detection and inter-sensor transfer learning on smart manufacturing datasets," 2022. [Online]. Available: https://arxiv.org/abs/2206.06355

[6] W.-P. M. Z. N. F.-C. A. C. J. Tabaszewski M, Twardowski P, "Machine learning approaches for monitoring of tool wear during grey cast-iron turning," june 2022. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35744419/

[7] E. Alegre, R. Aláiz, J. Barreiro, e. A. Viñuela, M.", and M. L. Cortés, "Tool insert wear classification using statistical descriptors and neuronal networks", booktitle="progress in pattern recognition, image analysis and applications," 2005. [Online]. Available: https://link.springer.com/chapter/10.1007/11578079_82

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, cite arxiv:1512.03385Comment: Tech report. [Online]. Available: http://arxiv.org/abs/1512.03385

[9] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact github repository." [Online]. Available: https://github.com/dbolya/yolact

## Appendix A
## Mask R-CNN Config

**GPU_COUNT = 1**, number of GPUs to use.

**BATCH_SIZE = IMAGES_PER_GPU * GPU_COUNT**, number of training samples used in one training iteration.

**IMAGE_MIN_DIM = 320, IMAGE_MAX_DIM = 320**, shape for scaling input images.

**LEARNING_RATE = 0.001**, step size at each iteration while moving toward the minimum of the loss function.

**RPN_ANCHOR_SCALES = (16, 32, 64, 128, 256)**, length of square anchor side in pixels. The size of predicted bounding boxes depends on this parameter, whose range of values allows to identify a variety of worn regions from small to large ones.

## Appendix B
## Baseline Mask R-CNN Training Config

In addition to parameters shown in Appendix A:

**IMAGES_PER_GPU = 4**, number of images to train with on each GPU.

**NUM_CLASSES = 4**, BACKGROUND, OK, DOUBT and NOK for the baseline model.

**STEPS_PER_EPOCH = xxx**, number of training steps per epoch. In our case, this is set to a variable number computed as $\left\lceil \frac{\text{Number of training samples}}{\text{Batch size}} \right\rceil$ to iterate once over the training set at each epoch. However, the choice is arbitrary.

## Appendix C
## Baseline Mask R-CNN Inference Config

In addition to parameters shown in Appendix A:

**IMAGES_PER_GPU = 1**, number of images to train with on each GPU. Evaluation is performed on one image at a time.

**NUM_CLASSES = 4**, BACKGROUND, OK, DOUBT and NOK for baseline model.

**DETECTION_MIN_CONFIDENCE = 0.7**, allows only predictions whose confidence is greater than or equal to DETECTION_MIN_CONFIDENCE to be returned as output.

## Appendix D
## Stage I - Mask R-CNN Training Config

In addition to parameters shown in Appendix A:

**IMAGES_PER_GPU = 4**, number of images to train with on each GPU.

**NUM_CLASSES = 2**, BACKGROUND and WEAR for stage 1.

**STEPS_PER_EPOCH = xxx**, number of training steps per epoch. In our case, this is set to a variable number computed as $\left\lceil \frac{\text{Number of training samples}}{\text{Batch size}} \right\rceil$ to iterate once over the training set at each epoch. However, the choice is arbitrary.

## Appendix E
## Stage I - Mask R-CNN Inference Config

In addition to parameters shown in Appendix A:

**IMAGES_PER_GPU = 1**, number of images to train with on each GPU. Evaluation is performed on one image at a time.

**NUM_CLASSES = 2**, BACKGROUND and WEAR for stage 1.

**DETECTION_MIN_CONFIDENCE = 0.8**, allows only predictions whose confidence is greater than or equal to DETECTION_MIN_CONFIDENCE to be returned as output.

## Appendix F
## Stage One Model - YOLACT Config

**LEARNING_RATE = 0.001**, step size at each iteration while moving toward the minimum of the loss function.

**BATCH_SIZE = 8**, number of training samples used in one training iteration.