Graduation Internship Report

Hilti Fast Falcon

Student:
Émilie Mathian

Professor:
Dr. Sergio Peignier

Supervisor:
Ing. Stefan Unger

August 23, 2020

Acknowledgment

First of all, I would like to thank **Hilti** to have allowed this great experience, despite the critical situation of this year 2020. I would like to particularly thank **Stefan Unger** to have supervised this project with ambition and realism, **Julian Koch**, for his valuable technical advice and **Bhargav Bhatt**, with whom I have had the pleasure to work all over the project. I want to thank the **INSA of Lyon** significantly for allowing me to acquire various technical and human skills, and therefore to have allowed the most efficient course of this experience. Especially, I would like to thank the department of **Bioinformatics and Modelisation** to have broadened our scientific culture to many subjects and fields of study that have all been used to solve this industrial case study. I want to take the opportunity to refer to our future head **Dr.Carole Knibbe** to have taught us the algorithmic with much pedagogy. I would also like to thank **Dr. Sergio Peignier** as the referring teacher of this internship, and for the quality of his various teaching. Finally, I would like to thank the **Unitech international** organization, which has allowed me to spend a wonderful year all over Europe, opening the doors of various universities, especially those of **Chalmers** (Sweden) and companies whose **Hilti** and **Geberit**.

Contents

1 Introduction **1**
- 1.1 About Hilti group . . . 1
- 1.2 Organization of the headquarters (Hilti Schaan) . . . 1
- 1.3 Presentation of the quality team . . . 2
- 1.4 Introduction to the case study . . . 2

2 Materials and methods **4**
- 2.1 Description of the case study . . . 4
- 2.2 Optical Character Recognition overview . . . 6
- 2.3 Object detection: state-of-the-art . . . 7
- 2.4 Description of Faster Region-based CNN (R-CNN) . . . 9
  - 2.4.1 Feature network CNN . . . 9
  - 2.4.2 Region Proposals Network (RPN) . . . 11
  - 2.4.3 Region-based convolutional network . . . 12
- 2.5 Training process . . . 13
- 2.6 Description of the data set . . . 14
- 2.7 Description of the software and hardware environment . . . 15
- 2.8 Metrics to evaluate the model's performances . . . 16
- 2.9 Blind comparison between the expected and the predicted label . . . 17
- 2.10 Detection of damaged characters . . . 18
- 2.11 Application: "Hilti Fast Falcon" . . . 19

3 Results **19**
- 3.1 Comparison of commercial OCR services . . . 19
- 3.2 Faster R-CNN: Comparison between two features extractor CNNs . . . 21
- 3.3 Detection of damaged characters . . . 24
- 3.4 Application: "Hilti Fast Falcon" . . . 24

4 Discussion **26**
- 4.1 Improving the model . . . 26
  - 4.1.1 Refine damaged characters detection . . . 26
  - 4.1.2 Refine the alignment between the predicted and expected labels . . . 29
- 4.2 Improving the application . . . 30

5 Conclusion **30**

References **31**

6 Supplementary figures **34**

7 Faster R-CNN Inception-v2-Resnet-v4 configuration **40**

Abstract

The applications of Machine Learning and Computer Vision are rapidly increasing in all fields of industry. It has undoubtedly affected the processes of quality assurance. In Schaan (Liechtenstein), Hilti AG is currently aiming to implement various Machine Learning algorithms, particularly on the production of screws and anchors. Like this, this work aims to develop a text recognition application to automatically read and assess the quality of engravings on steel surfaces. Currently, despite manual visual inspections carried out, false and unreadable embossments of screws and anchors are not always recognized. In the last years, the losses to the company because of such errors are worth tens of thousands of Swiss Francs. Incorrect or defective embossing on screw and anchors should be identified so that the punch can be replaced as soon as possible. To reach this goal, we implemented an object detection network grounded on the Faster R-CNN algorithm, which allows reading curved embossing text with an accuracy higher than 99%, whatever the orientation of the object. This model has been integrated into a graphical user interface remotely connected to a GPU playing the role of a server, to keep the detection time as short as possible. The solution developed is already working on the production line. We are currently improving the detection of the wear of punches through the semantic segmentation done by Mask-RCNN and extending the model to more products.

1 Introduction

1.1 About Hilti group

Hilti was founded in 1941, in Lichtenstein, by Martin Hilti. Since its creation, this familial company has become an international leader in the construction industry, with high-quality tools and technologies. Today the company counts 30000 employees, whose 15000 in the sales force, and nine plants in Schaan (Lichtenstein - headquarters), Thüringen (Austria), Kaufering and Strass (Germany), Kecskemet (Hungary), Zhanjiang (China), Matamoros (Mexico) and Gujarat (India). The current CEO, Christian Loss, introduced the "Champion 2020"[1] corporate strategy. Focusing their strengths on product and services differentiation and a direct customer relationship, Hilti improved its results; therefore, this winning strategy has been extended to 2023.

Hilti's direct sales model is one of its main strengths in front of its competitors. Hilti products are dedicated to professionals, who can buy them directly in Hilti's stores on their website, or by telephone. Their close relationship with construction professionals, and their presence on building site, increase their innovation capacities. Like this, the company invests 6 percent of its turnover each year in research in development, allowing it to stay ahead in terms of technology [2].

If Hilti main products remain, anchors, installation systems, fire protection installation, power tools, measuring and detection tools, demolition hammers, cordless electric drill, and diamond drills, the company is also diversifying its activity toward services like Building information and modeling (BIM) services, or design software. This internship took place in Schaan in the context of the transition toward industry 4.0, focusing on how artificial intelligence (Artificial Intelligence (AI)) techniques can be used to increase production quality. [4]

1.2 Organization of the headquarters (Hilti Schaan)

Customers' needs and requirements define the global strategy of the whole company. Customers interact directly with the sales department, which gathered around 50% of Hilti's employees. The information collected by the sale department help to define the objectives of the different Business Unit (BU). Hilti's business units are divided into two business areas: the area of electric tools and accessories and the area of fastening and protection. Each BU represents a division of the global company and is responsible for a set of products and services that belong to a specific field of activity. Each BU must define the strategy and make operational decisions according to the specific goals and issues of

[1] https://hilti.sharepoint.com/sites/redi/Companypages/Pages/CorporatePages/knQGvcd9NLQ7ZLtwhJaG9A/3226cf28-a00b-4c05-8a69-16bf937410af.aspx

[2] https://www.hilti.group/content/hilti/CP/XX/en/company/corporate-information/Strategy/our-business-model.html

[4] https://www.hilti.group/content/hilti/CP/XX/en/company/corporate-information/company-profile.html

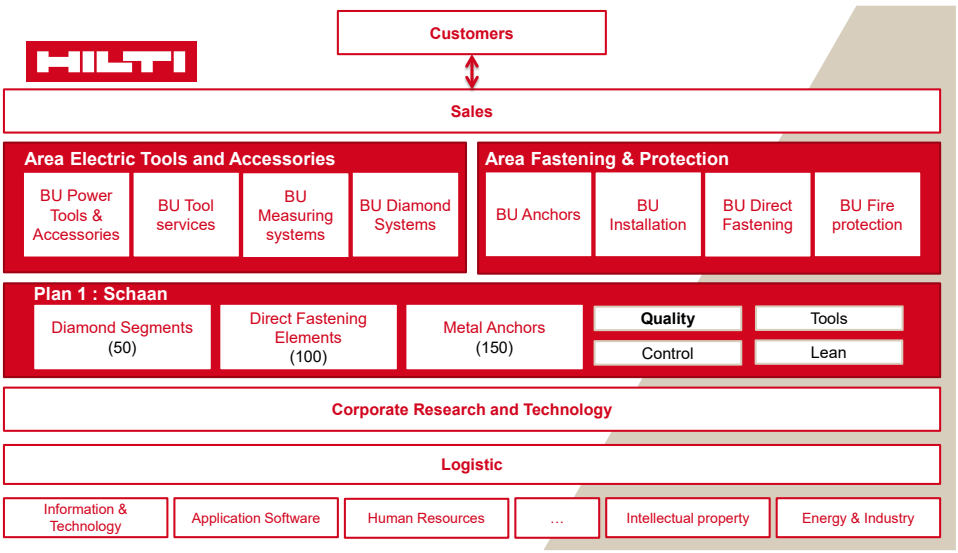


Figure 1: **Organisation of Hilti Schaan.** Diagram presenting the main departments and their interactions[3].

their respective field. The organization and objectives of plans result from these BUs' strategies. The production activities in Schaan are divided between diamond segments, direct fastening elements, and metal anchors. The production lines are coordinated and supported by the control, the quality, the lean, and the tools[5] units. This internship has been realized within the quality team, introduced in the following section. Customers' expectations and BUs' strategies define the research and development (R&D) goals. The budget annually allocated to the R&D is invested at 30% in long term differentiation projects and respectively 50% and 20% in mid and short term projects[6]. Finally, the organization relies on global departments such as information and technology, human resources, intellectual property [Fig.1]...

1.3 Presentation of the quality team

The quality team's activities are under the direction of the global manufacturing department lead by Armin Kueper. Seppo Paraemaeki, the head of the quality, leads a team of 22 people whose my supervisors Unger Stefan (Quality manager) and Koch Julian (Quality specialist), and my colleague Bhargav Bhatt (Intern in computer vision).

1.4 Introduction to the case study

Machine learning and artificial intelligence techniques catch more and more attention in various fields of application (industrial, medical, etc.). Presently they can be seen above all in the networking of machines and the emergence of novel technology and systems that

[5]The tools unit produces the elements needed for the production of the final products.

[6]https://www.hilti.group/content/hilti/CP/XX/en/company/corporate-information/company-profile/company-profile.html

can significantly increase productivity, efficiency, and quality within production. These methods include predictive maintenance solutions, independent manufacturing processes, or automated quality controls. Especially, deep learning models can be used to support manual visual inspection tasks. For some authors, these systems detect a higher defect rate than comparable inspection performed by humans [1]. Following these inclinations, the research paper within the Smart Factory Initiative of Hilti AG focused on developing a visual inspection system that can be used for screws and anchors. Therefore, this internship aims to implement implementing a system that can read and assess the quality of the engraving text on steel surfaces.e. Our research is, therefore, focused on the following questions:

1. In which cases can a commercial Optical Character Recognition (OCR) software, knowing that it is the quickest solution in terms of implementation and the easiest in terms of maintenance? This question implies determining which is the best commercial service for our case study.
2. Since these commercial OCRs services are not efficient in reading curved texts [2]; we have focused our attention on object detection techniques [3]. Then the central question of this study is: which is the object detection technique allowing reaching the best accuracy [4, 5, 6, 7]? The training of an object detection network with the goal to reach 99% of entirely correctly annotated pictures brought many challenges. Firstly, the training set must be wide enough to cover most cases and notably different object orientations. The size of the training set led us to think about a semi-supervised labeling process, to shorten this stage extremely time-consuming. Secondly, the performances obtained in the laboratory have to been transposable on the production line, and therefore a strict and reproducible framework has to be defined.
3. Finally, since the solution has to help the workers in their daily tasks, we have to think about an application user-friendly in terms of use and speed. These led us to question ourselves about the best software architecture guaranteeing both stability and speed.

This report aims to answer these questions. That is why we first developed a methodological section that described in detail the study case and its challenges, the global functioning of commercial OCR services, and object detection techniques. We established a review of the existing object detection algorithms focusing our analysis on Faster R-CNN, the most accurate method according to the literature [5]. This section also gathers the different steps that allowed the setting up of this model, namely the composition of the data set and a description of the labeling process, the descriptions of the soft-wares and hard-wares environments, the optimization algorithm chosen, and the metrics that allowed us to assess the performances of the models. Finally, this section describes how we implemented a user-orientated solution with a description of the protocol comparing the expected label

to the predicted one, and a description of the Graphical User Interface (GUI) architecture. The results section firstly presents a comparison between three commercial OCR services Google [8], Azure [9], and AWS [10]. Secondly, it presents our object detection algorithm's performances, including a comparison between two feature network Convolutional Neural Network (CNN). Finally, the results section included a detailed description of the user's interface and the underlying Multi-clients/Server architecture. These results allowed us to open a discussion on users' feedbacks and the following objectives in this case study, especially on how to extend the existing solution to other products and how to improve the detection of the wear of the stamps[7].

2 Materials and methods

2.1 Description of the case study

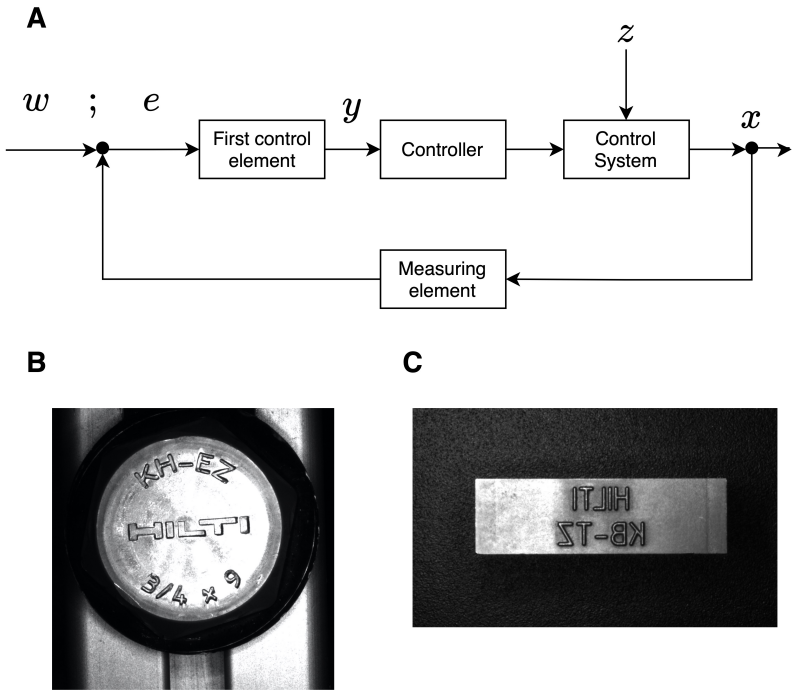


Figure 2: **Presentation of the problematic. A)** Quality control cycle diagram [11].**B)** Typical picture of screw [*]. **C)** Typical picture of stamp for anchor labeling [*].

We worked on the quality control of screw[8] [Fig.2b] and anchors labeling [Fig.2c]; in the following section, we will use a model named quality control cycle to describe the problem [Fig.2a]. This model, based on the technical control circuit, aims to create a product with a predetermined quality [11]. Therefore, the quality control cycle consists of an activity's execution, the analysis of the resulting quality according to measurable

[7]Here, we call stamps the metallic tools used to punched the anchors and screws.

[8]In this report, we refer to the "KH-EZ" and "KH" Hilti's screws used for permanent fastening in concrete.

characteristics values, and possible triggering of corrective actions. Currently, the quality of the labeling, refers as x the control variable [Fig.2a], is checked by visual inspection. Two reasons can mainly explain the deviation e, between the quality required w and x:

1. the wrong stamp has been used;
2. the wear of the stamp during the production led to the quality deterioration of the characters punched.

These hazards z, at the origin of e, have to been captured by a measuring element. Since the current quality control cycle is grounded on visual control, the measuring element is the employees' eyes. As the first control element, workers compare the quality of the produced label to the expected one. Nevertheless, this crucial comparison is subject to failures. Effectively, the wrong stamp's use leads to losses of several thousand Swiss francs in the past years. It can be explained because production planning, care is taken to ensure that similar anchors and screws are produced after the other so that the setup time is kept as short as possible. However, these products have similar designations, which increases the risk of using the wrong stamp. This case can also be explained by the fact that stamps show a mirror-inverted profile that complicates the reading [Fig.2c]. The detection of the wear of the stamp, in the current setup, is a subjective measure, that can differ between workers, and from time to time. Mainly, it has been noticed that characters with closed forms such as 0, 8, or 9 are more sensitive to this issue, because, during the embossing, the material inside the closed areas cannot spread outwards and thus pressed against the punch. To reduce those errors, the employees have been asked to copy down the embossing during the visual inspection and compare it by a second worker [Fig.S1a].

Despite these precautions, expansive errors remain; that is why we will propose a quality control cycle grounded on computer vision techniques. The measuring element becomes a camera, and the first control element a neural network or an OCR software, to predict the characters on pictures. The return variables of these algorithms are probabilities reflecting the degree of confidence in the prediction for each character. The characters associated with high probabilities are used to generate a string (the predicted label) compared to the expected one [Fig.S1b]. Finally, in these two cycles, the only measures sufficient to restart the production are to clear or replace the stamp [Fig.S1].

Our case study is focused on two types of embossing with the linear and curved cases. For the linear engravings used for anchors, an existing OCR service will be used. If this solution is fast to implement and easy to maintain, it has proven to be not able to read the curved text [2]; that is why for screw labels, we implemented our object detection network. In this case study, several challenges have to be taken into account. Firstly, picture quality should be constant and as closed as possible than the one used to trained the network. Thus, despite the varying lighting conditions in the production line, objects should be illuminated continuously to eliminate the potential reflections induced by the

steel surface. Secondly, the algorithm response time has to be kept as low as possible to encourage the use of the new solution. Thirdly, the solution has to be compatible with company policies and therefore develop under Microsoft environment. Fourthly, it should be easily generalizable for other types of anchors and screws.

2.2 Optical Character Recognition overview

OCR algorithms work in mainly three steps [2]:

1. Localization and detection of the text,
2. Enhancement and segmentation of the character,
3. Classification of the detected characters.

Mainly three approaches are used to realize the first step, which is crucial for text detection in natural scenes. It can be done by edges detection methods that apply mainly canny filters and morphological operations to extract text area. Nevertheless, these techniques are sensitive to shadow, highlighting, and too complicated backgrounds. The second approach consists of texture analysis through Gaussian filters, Fourier transformation, or local binary pattern analysis. The extracted regions are then used to feed a binary classifier grounded on a neural network, or a different heuristic, to discriminate text and non-text areas. The third technique is based on connected-component based methods [7]. Firstly the picture is binarized, and the smallest and largest components (*i.e.*, regions in the binary picture) are deleted. Then, neighboring components sharing a similar intensity histogram are merge based on the assumption that text components shared more similar intensity histograms than non-text components. Finally, to prune non-text areas, the affinity within components is measured through the standard deviation between the inner distance between all pixels in a component. Assuming that text components are composed of different characters, they have a higher affinity; like this, the components with an affinity lower than a given threshold are classified as background and removed [7]. This last technique is often more accurate than the others since it focuses on global and detailed pixels analysis [2]. The second step consists of text enhancement and segmentation. Whose one of the more difficult steps is picture binarization, since finding the right threshold is a problem on its own. Different methods can be applied, such as dichotomic binarization per region, defining themselves by pixel intensity, or adaptative thresholding methods based on texture analysis [2]. Then, text orientation should be corrected, and so pictures' skew angle has to be calculated. Several techniques are commonly used, such as the projection profile. This method assumes that a based line is mainly defined by black pixels, like this, vertical and horizontal lines are projected to get the sum of the black pixels that they overlap. The image is rotated degree by degree, and the sum is recalculated, if it is higher than the current one, the score is updated. After repeating the operation until rotation of 90°, the maximal score can be associated with the picture's

skew angle. Another approach consists of searching the nearest neighbors of each character and computing the histogram of these values to determine the pictures' skew angle [12]. Text enhancement also consists of image despeckle, applying different filters such as Gaussian and blur filters. Finally, the text is decomposed into lines, lines into words, and words into characters. This segmentation process is done by detecting the gaps in the picture's binary histogram, since gaps' length and height are good indicators of spaces between lines, words, and finally, characters. The final step involves feeding a neural network trained to classify the characters, with the crop pictures isolating the presumed characters. This last step can also be realized through a matching profile technique, but it has been progressively left for neural networks to deal with more various font sizes. OCR services are often efficient on scanned documents, but extracting text in natural scenes is a highly more challenging problem. The software has to deal with more various font sizes, different text orientation, and alignment, potential distortions resulting from camera angle and perspective, or weak edges between the text and the background [2]. In this case study, we compared three commercial OCR services: Google OCR [8], Azure OCR [9], and AWS OCR [10]. Their performances have been respectively assessed on linear stamp and screws cases [9].

2.3 Object detection: state-of-the-art

Convolutional neural networks (CNN) are the most used and the highest-performing systems for object detection problems. Object detection is a computer vision problematic consisting in determining the position and the class of each object in a picture. A naive approach would involve segmenting a picture in windows to feed vanilla classifier that would predict for each fragment the presence or absence of an object and its potential class. This time-consuming approach implies that the size of the different objects is known and constant [13]. Since this assumption cannot generally be made most object detection methods propose adaptive size bounding boxes to surround proposals. In the following section, we will present the state-of-the-art object detection techniques and then describe the most accurate one Faster R-CNN.

Object detection techniques based on CNN, have been introduced with the R-CNN algorithm, for Region proposals with CNN[4]. R-CNN uses a selective search algorithm to extract regions of interest (Region of Interest (RoI)), that potentially contain objects [Fig.3a]. In order to do so, the selective search algorithm segments images based on pixels' texture and colors similarity [Fig.3a-step 2]. Then the regions sharing the same features are agglomerated to create larger areas according to regions' size and shapes compatibility [14]. After resizing the resultant region proposals, these are successively used to feed a CNN, from which fixed-length feature vectors are extracted [Fig.3a-steps 3-4]. The resulting feature vectors are given as input to different linear support vector

[9]This section explains the general functioning of OCR services since, the technical details of the commercial soft wares compared are not available.

machine (SVM) models, trained individually to classified the different classes of objects [Fig.3a-step 5]. Finally, a linear regression model is used to adjust the coordinate of the predicted bounding boxes [Fig.3a-step 5'].

The author of R-CNN developed a second version of R-CNN named Fast R-CNN, which allows a reduction of the computation time [3] [Fig.3b]. A deep CNN, such as VGG takes as input the entire image and creates as output a feature map [Fig.3b-step 2]. The regions of interest (ROIs), extracted through the same selective search algorithm as previously described [14] [Fig.3b-step 1'], are projected on these features map. After resizing each RoI through a max-pooling layer [Fig.3b-step 3], they are successively used as input for fully connected layers (Fully connected layer (FC)) [Fig.3b-step 4]. The network has two outputs, a first FC layer with a softmax activation function returns the probabilities for an object to belong to each class [Fig.3b-step 5], and a second FC is used as a category-specific bounding boxes regressors [Fig.3b-step 5']. The better performances of Fast R-CNN can be explained by the fact that instead of feeding a CNN successively with the different proposals, the deep CNN is trained on the full picture to generate the feature map [5].

Faster R-CNN is the latest version of object detection based on R-CNN techniques, which is also the fastest [Fig.3c] [5]. As Fast R-CNN Faster R-CNN used a deep CNN to generate a feature map. However, instead of using the greedy selective search algorithm to generate ROIs, Faster R-CNN is composed of a second CNN, called region proposals network. The detailed architecture of Faster R-CNN will be explained in the following section.

Finally, two other methods names respectively You only look once (YOLO) [Fig.3e] [6], and Single-shot detector (SSD) [Fig.3d] [15], do not use region proposals but a single feed-forward convolutional network. Both YOLO and SSD segment the input picture into a grid of size $S \times S$, and for each cell B predictions corresponding to B different predefined bounding boxes, also named anchors are made. Each prediction gathers the probability that the box contains an object and its coordinates defined with four values x, y, w, and h, where x and y correspond to the center of the box, and w and h correspond respectively to its width and its height. Like this, a model trained on N_c classes has an output size of $S \times S(B \times 5 + N_c)$. The main difference between YOLO and SSD is that YOLO uses two FC layers after the main CNN [Fig.3e-E Step 3], whereas SSD uses various feature maps resulting from different convolutional layers of the main CNN [Fig.3d-step 3-3'-3"] to generate the final predictions. YOLO is more accurate than SSD, which is faster [16]. Although these two models are faster than the region-based methods, they are less accurate [16], so we did not explore them to focus our attention on Faster R-CNN [5].

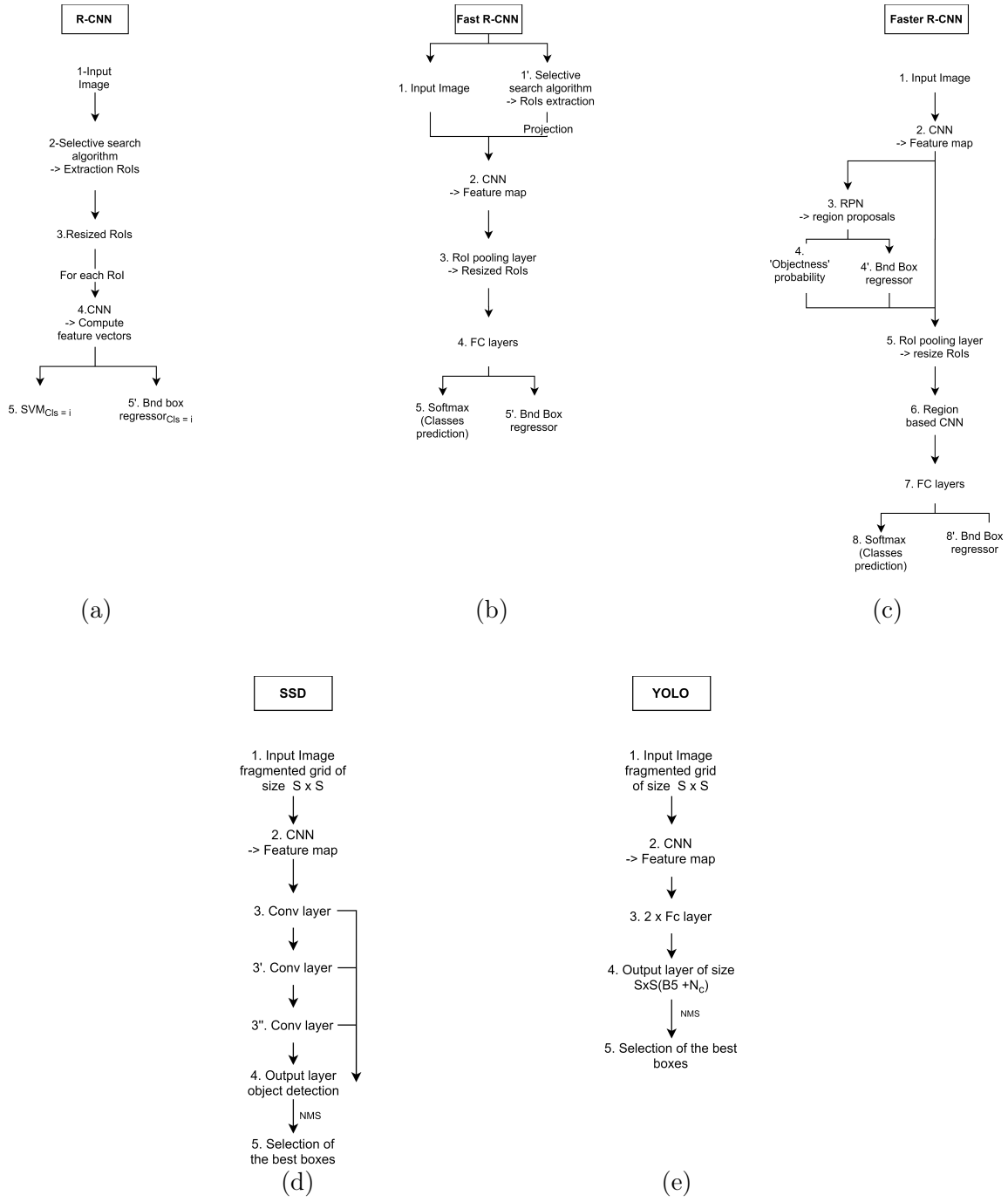


Figure 3: **Object detection techniques based on CNNs [*].** The following diagrams summarizes the main steps of the following object detection techniques:**A)**R-CNN [4], **B)** Fast R-CNN [3], **C)** Faster R-CNN [5], **D)** Faster R-CNN [5], **E)** YOLO [6], where S refers to the grid, B refers to the bounding boxes and Nc to the number of classes (the other abbreviations are listed in the index).

2.4 Description of Faster R-CNN

2.4.1 Feature network CNN

Different feature extractors have been implemented for Faster R-CNN. We chose Inception-v4-Resnet-v2 [17] as a feature extractorCNN after comparing it with Inception-v2 [18] (6).

Therefore, in the following section, we will describe described its features.

Inception:

Inception modules have been developed to answer information distribution issues; larger kernels should be used for globally distributed information and reciprocally for local information. These filters operate on the same level to not increase the deepness of the network, which is prone to over-fitting and vanishing gradient issues. Given that convolutional operations are computationally expansive, 1×1 filters are applied at modules' entrance, reducing the number of input channels [Fig.4-b-c-d- Turquoise boxes] [19]. Filter bank expansions are not only computationally cheaper, but they also allow to avoid the representational bottleneck, *i.e.*, drastically dimension reduction of the input data [Fig.4-b-c-d- Turquoise boxes]. Furthermore, to improve the convolutional speed, these operations are factorized. For example, the convolutional operation using a filter of size 7×7 is decomposed in two successive operations using filters of size 1×7 and 7×1 [18] [Fig.4c Orange boxes]. Contrary to the previous implementation of Inception, Inception-v4 [17] uses more uniform modules allowing to boost the performances [Fig.4-a Green boxes]. This implementation also used a different upstream network to the Inception modules [Fig.4-a Blue box] and reduction blocks modifying the width and the height of the grid [17].

Resnet:

Much empirical evidence showed that deeper CNNs could learn more complex data, justifying the enthusiasm around Inception modules. Nevertheless, deeper networks can lead to the degradation of training accuracy [20]. Deeper networks are more difficult to optimize and generally suffer from exploding/vanishing gradient problem. ResNet [20] connections solve this issue by introducing a "deep residual learning framework". Let us consider $H(X)$ as an underlying mapping of a few stack layers with X denoting the input. We define the residual function $F(X)$ such as $F(X) \coloneqq H(X) - X$, like this the original function to optimize become $F(X) + X$ [Fig.4-e]. This reformulation solves the degradation problem. Supposing the optimal solution is the identity mapping. It will be easier to set the residual to zero than learn an identity mapping by multiple non-linear operations. In consequence, a residual network learns the perturbation with the identity mapping as reference. Since X and $F(X)$ must have the same dimension 1×1 convolutional operations are applied after the original operations [Fig.4-b-c-d Red elements]. Furthermore, to respect this propriety, the identity mapping should be multiplied by a linear projection W; thus, the output y becomes $:F(X, W_i) + W_s X$, where W_i is the weights matrix, and W_s the linear projection.

Residual connections integrated with Inception-v4 allows reaching quicker a higher accuracy [Fig.6-b-c]. The purpose of the feature network defined above is to extract an intermediate convolutional layer (*Mixed-6-a*) to build feature a map of dimension $D \times D$ [Fig.3c-step 2], that is used to generate region proposals [Fig.3c-C step 3].

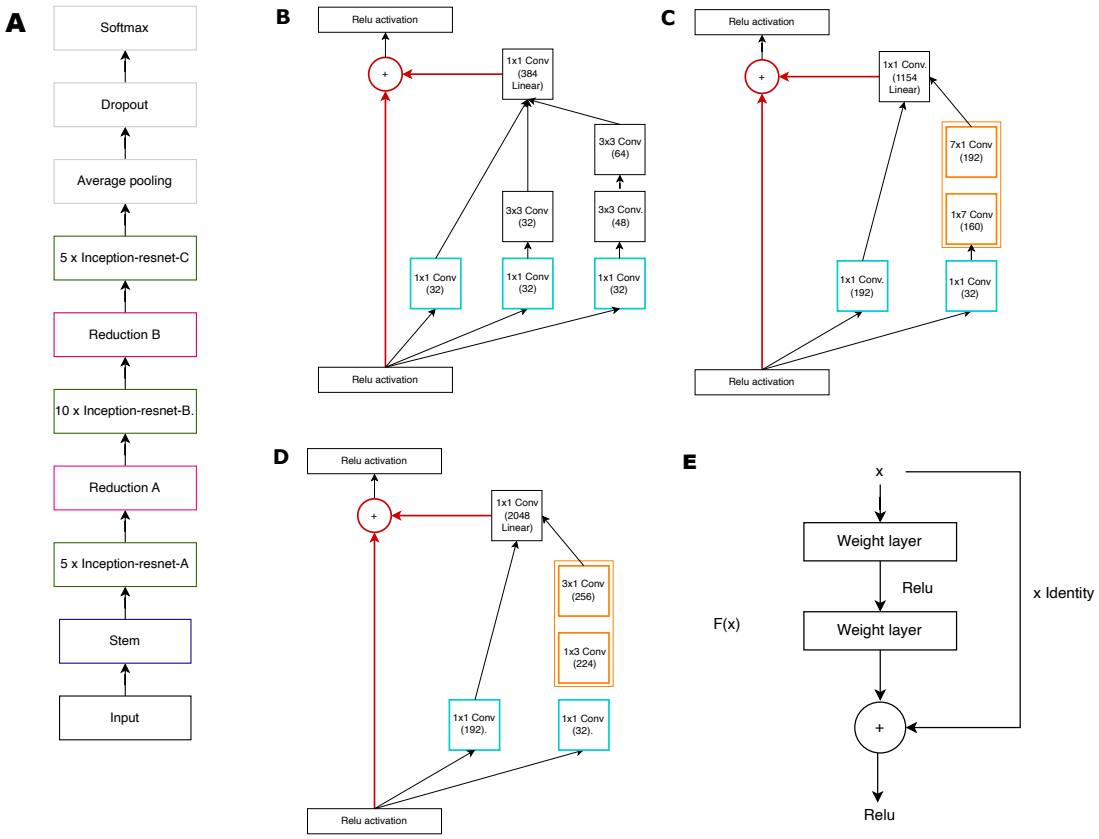


Figure 4: **Architecture's details of the feature extractor network Inception-v4-Resnet-v2 [17] A)** The global architecture of Inception-v4-Resnet-v2. The gray layers are not used in the Faster R-CNN model since an intermediate convolutional layer is extracted to produced the feature map. **B)** Detail of the block Inception-ResNet A. **C)** Detail of the block Inception-ResNet B. **D)** Detail of the block Inception-Resnet C. **E)** Central dogma of the residual connections.

2.4.2 Region Proposals Network (RPN)

For each point in the feature map, a set of k reference bounding boxes named anchors is calculated. These anchors are themselves defined by the coordinate of their center (x, y), their width w, and their height h. To handle objects of different sizes, this set of anchors sizes gather boxes of different ratios and sizes. The Region Proposal Network (RPN) takes as input windows of size $n \times n$ extracted from the feature map. They are mapped on a lower-dimensional matrix though a convectional layer [Fig.3c-step 3]. Then this matrix fed two parallel convolutional layers. One, with an output of size $2k$, returns for each anchor the probability that it contains an object ("objectness score") [Fig.3c-step 4]. The second one, with an output of size equivalent to $4k$, allows adjusting bounding boxes' coordinates [Fig.3c-step 4'].

Each anchor is categorized as a foreground region, if its Intersection over Union (IoU)[10] with a ground-truth object is higher than 0.7, and as a background region if this ratio

[10]IoU for Intersection over Union is the ratio of the area shared between two bounding boxes over the area of their union.

is lower than 0.3. (The anchors with intermediate IoU values are not considered.) with a balanced ratio between the ones classified as foreground and background, is generated. This sampling allows calculating the multi-task loss function is defined such as

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{1}$$

where for the anchor i, p_i refers to its "objectness score" and p_i^* is equivalent to 1 if the anchor is classified as a foreground element and 0 otherwise. The classification loss (L_{Cls}) is weighted by N_{Cls}, which corresponds to the number of anchors in the mini-batch. The second term of the equation (1) represents the regression loss (L_{reg}), where t_i is a 4-size vector representing the coordinate of the predicted bounding box, and t_i^* corresponds to the coordinates of the ground-truth bounding box. Given the definition of p_i^* only the anchors classified as foreground elements are taken into account. The total number of anchors weights this second term (*i.e.*, $N_{reg} = D \times D \times k$). Finally, the parameter λ allows giving weight to the regression part of the loss function. In more detail, the regression loss is calculated, such as $L_{Reg} = R(t_i - t_i^*)$, where R is the $smooth_L1$ function[11], allowing to consider as almost correct the bounding boxes with small error. Taking as an example the regression loss calculation on the x coordinate, t_x and t_x^* are defined respectively, such as $t_x = (x - x_a)/w_a$ and $t_x^* = (x^* - x_a)/w_a$, where x, x_a, and x^* correspond respectively to the coordinates of the predicted bounding box, to those of the anchor and those of the ground-truth bounding box. Since the anchors overlap, the proposal regions end up also overlapping the same object; therefore, a Non-Maximal Suppression (NMS) algorithm [21] is used to delete the predicted boxes whose the IoU with other predicted boxes is above a given threshold. Finally, the top N_f proposals are kept to be classified.

2.4.3 Region-based convolutional network

Each of the N_f proposals extracted from the feature map is resized owing to a max-pooling layer [Fig.3c-step 5] to get a fixed size matrix. Each of these matrices feeds a Region-based CNN [Fig.3c-step 6]. This last network has for objectives to assign a class to each proposal and to adjust their coordinates, according to objects' class. In this goal, the fixed-size matrices are firstly flattened, the resulting vector then feeds two FC layers [Fig.3c-step 7]. One is used to predict the object's class, and so it is composed of $\#Classes + 1$ units, where $Classes$ represents the total number of classes in the model more 1 for the background [Fig.3c-step 8]. The second one is used to adjust bounding boxes coordinates, and so it has an output size equivalent to $4 \times \#Classes$ [Fig.3c-step 8'].

[11]

$$Smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1. \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{2}$$

2.5 Training process

Training a network means to present repetitively input matrices (pictures) for which the output values are known. This allows calculating the loss function, whose value represents the distance between the predicted and the expected outputs. Therefore, training a network is an optimization problem, consisting of finding the weights that minimize the loss function. This optimization process is done through a gradient descent algorithm [22]. Gradient descent methods allow minimizing a cost function $J(\Theta)$, where Θ is the model's parameters. Like this, the optimization consists of updating the parameters in the opposite direction of the gradient $\nabla_\Theta J(\Theta)$. The size of the step to reach a minimum is called the learning rate η. The most basic optimizer is the vanilla gradient descent algorithm that computes $\nabla_\Theta J(\Theta)$ for the entire data set. Therefore, the parameters' update is defined by the following equation:

$$\Theta = \Theta - \eta.\nabla_\Theta J(\Theta). \tag{3}$$

Nevertheless, this algorithm is intractable for large training sets, and suffers from the following drawbacks [22]:

- Tuning η is an important and difficult process. If η is too small the convergence will be exaggeratedly too slow if it is too large $J(\Theta)$ will fluctuate around the minimum;
- $J(\Theta)$ is often a highly non-convex function; that is why a vanilla gradient descent will often get trap in a local minimum.

Those issues can be overcome through a stochastic Gradient Descent (SGD) optimizer. This method updates the parameters for each training example. For an input matrix $x^{(i)}$ and an output vector $y^{(i)}$, the update equation is:

$$\Theta = \Theta - \eta\nabla_\Theta J(\Theta; x^{(i)}; y^{(i)}). \tag{4}$$

SGD performs frequent updates that induced high fluctuations in the optimization process. These fluctuations allow escaping local minima but complicate the convergence to the global minimum. Furthermore, a SGD has trouble navigating around very steep minima and oscillating across the ravines' slope [22]. That is why the training process has been realized using the SGD algorithm with momentum γ. An SGD with momentum, adds to each update u_t a fraction γ of the previous update u_{t-1}. Like this, the following equation describes the parameters' update:

$$u_t = \gamma u_{t-1} + \eta\nabla_\Theta J(\Theta), \tag{5}$$

$$\Theta = \Theta - u_t. \tag{6}$$

Like this, the update process is described by the following logic:

$$
\begin{aligned}
&u_0 = 0 \\
&u_1 = \gamma u_0 + \eta \nabla_{\Theta_1} J(\Theta_1) = \eta \nabla_{\Theta_1} J(\Theta_1) \\
&u_2 = \gamma u_1 + \eta \nabla_{\Theta_2} J(\Theta_2) = \gamma\eta \nabla_{\Theta_1} J(\Theta_1) + \eta \nabla_{\Theta_2} J(\Theta_2) \\
&u_3 = \gamma u_2 + \eta \nabla_{\Theta_3} J(\Theta_3) = \gamma(\gamma\eta \nabla_{\Theta_1} J(\Theta_1) + \eta \nabla_{\Theta_2} J(\Theta_2)) + \eta \nabla_{\Theta_3} J(\Theta_3) \\
&\quad ... \\
&u_t = \gamma u_{t-1} + \eta \nabla_{\Theta_t}) J(\Theta_t) = \gamma^{t-1} \eta \nabla_{\Theta_1} J(\Theta_1) + \gamma^{t-2} \eta \nabla_{\Theta_2} J(\Theta_2) + ... + \eta \nabla_{\Theta_t} J(\Theta_t).
\end{aligned}
$$

In analogy with a ball descending a hill, accumulating momentum and so going faster and faster, the momentum term increases when gradients point in the same direction and decreases when gradients change of direction. In consequence, the convergence is faster, and with fewer oscillations [22]. For the training, we set $\gamma = 0.9$.
We trained Faster R-CNN Inception-v4-Resnet-v2 for 140000 steps ($\simeq$ 14.4 epoch). The training time has been defined *a posteriori* according to the convergence of the training and the validation loss curves. We tuned model's parameters according to Faster R-CNN MSCOCO dataset [23] configuration [12], the main modifications are the following:

- Number of classes = 20;
- Batch size = 1;
- Image resizer: min dimension = 600, max dimension = 600;
- Initial learning rate = 0.003;
- queue capacity = 2, min after dequeue = 1, num reader = 1[13].

The full configuration file is joined in annex [Annex.7].

2.6 Description of the data set

Our application works for a set of screws of diameter, 3/4, and 5/8 inches. For each product, here a product refers to a screw with a particular label, several objects around five have been randomly selected. For each object, identified by an ID, a video of 5 to 7 seconds has been recorded. To reduce the noises, these videos have been realized under a constant powerful artificial light, and width a constant distance between the lens and the object. The screws have been rotated with an angle of $\pm 120°$, according to the 'natural' reading orientation taken as reference (0°). This rotation step fulfills two purposes:

1. producing data augmentation,

[12] https://github.com/tensorflow/models/blob/master/research/\object_detection/samples/configs/faster_rcnn_inception_resnet_v2_atrous_coco.config

[13] These parameters allow handling the quantity of data load at once in the GPU, to prevent memory overflow issue.

2. allowing a learning independent of text orientation, giving more freedom to users concerning the position of the object under the camera.

These videos have been then segmented to generate around 500 to 600 frames per screw ID. Each frame has to be labeled, *i.e.*, associating to an XML file containing the class and each characters' position. For each product, 50 to 100 pictures have been labeled manually with the GUI labelImg [14]. This long and tedious process has been shortened owing to a semi-supervised learning protocol. This procedure involves labeling a small number of pictures manually and extending the Faster R-CNN training using the latest frozen graph by transfer learning. For the labeling purpose, the training set contains all the pictures already labeled from the previous experiments and the ones that have just been manually annotated, and the test set the pictures to annotate. Each element in the test set is predicted using the frozen graph of the extended trained model. Predictions' correctness is determined comparing the pictures' filename, containing the expected label, and the class of the bounding boxes predicted, once they had been ordered. This process allowed to re-labeled manually only the picture incorrectly predicted. This protocol could have been used successfully because:

- The pictures database has been constructed progressively, therefore aside from the first experiment the latest frozen graph could have been used to train a new network by transfer learning;
- The same screw IDs belong to the training and the testing set; this voluntary redundancy eases the labeling process.

After labeling all the database, a fairer training has been launched. Each screw ID belongs either to the training set or to test and/or to the validation set, allowing a more consistent evaluation of the model's performances. For this final training, the proportional size of the training, the test, and the validation sets are respectively equivalent to 0.8, 0.05, and 0.15. This split obeys to following rules:

- Screws' pictures with the same ID cannot belong to the training set and simultaneously to the validation and/or to the test sets;
- As far as possible, a balanced ratio of each character is maintained between the three sets.

2.7 Description of the software and hardware environment

Hardware: We worked under Windows 10, with a Lenovo ThinkPad (Intel Core i5 gen 7th). To boost, We had a powerful external Graphics Processing Unit (GPU) with the following characteristics:

[14] https://github.com/tzutalin/labelImg

Model	Razor GEFORCE RTX 2080 TI GAMING X TRIO
GPU	Nvidia GetForce RTX 2080Ti
Architecture	Nvidia Turing
Boost Clock	1545 MHz
Cuda Core	4352
Memory size	11 GB
Memory bus	352 Bit

Table 1: external GPU features.

The videos have been recorded with a Basler ace acA1920-150um Monochrome USB 3.0 camera, installed on a support to guarantee a constant distance between the lens and the object. This support is also equipped with a lighting system to prevent potential noises resulting from reflections.

Software: We used the official Tensorflow object API repository[15] to create our training platform. This API offers an implementation of Faster R-CNN, with Inception-v4-Resnet-v2 as a feature extractor. We used Tensorboard API[16] to follow the progress of the training and the validation. We worked under a Python (3.7) virtual environment whose main libraries are Tensorflow-gpu = 1.15 and OpenCV = 4.2. To use the GPU, we installed CUDA Toolkit 10.0[17] and Cudnn 7.6.4 [18]. The GUI has been using PyQT5[19]. Finally, the Multi-clients/Server architecture has been implemented through the TCP/IP protocol using the basic Python Socket library 5[20].

2.8 Metrics to evaluate the model's performances

During the training, to evaluate the model's performances, we followed the metrics show by Tensorboard, *i.e.*, the loss curves [Fig.6-a], the mean Average Precision (mAP) [Fig.6-b] and Average Recall (AR) [Fig.6-c].At the end of the training, XML files, reporting for each bounding box predicted its location, class, and accuracy, have been generated for the all data set. The predicted XML files have been compared to the expected ones according to the following protocol:

1. For each bounding box i predicted we search in the expected XML file the bounding box j with the maximal IoU score.
2. If $IoU_{i,j} > 0.7$, i is classified as a True Positive (TP), and as a False Positive (FP) otherwise.

[15] https://github.com/tensorflow/models/tree/master/research/object_detection
[16] https://github.com/tensorflow/tensorboard
[17] https://developer.nvidia.com/cuda-10.0-download-archive
[18] https://developer.nvidia.com/rdp/cudnn-archive
[19] https://wiki.python.org/moin/PyQt
[20] https://docs.python.org/3/library/socket.html

3. Furthermore, if $class_i = class_j$, i is labeled as well-classified otherwise as miss-classified.
4. Each expected bounding box not found in the expected XML file is classified as False Negative (FN).

This process applied successfully to the train, the validation and the test sets allow generating the following metrics:

- Global metrics:
  - The proportion of images entirely correctly predicted;
  - The total proportion of character correctly predicted;
  - The proportion of images per screw labeled correctly predicted, to catch if one product has not been learned [Fig.6-d-f];
  - The proportion of images per screw ID correctly predicted, as an indicator of potentially poor quality picture associated with an object.
- Bounding boxes metrics:
  - The total proportions of TP, FP, and FN.
- Character metrics:
  - The recall per character, to asses model's sensitivity;
  - The precision per character;
  - The different errors occurrences [Fig.6-e-g];
  - The accuracy distributions per character [Fig.S5].

Accuracy distributions were used to assess character readability. Effectively, we noticed that despite very narrow distributions centered on 0.99, these metrics could be used to generate warnings.

2.9 Blind comparison between the expected and the predicted label

When a new picture is predicted, the question around comparing the expected and predicted text appears trivial, but not because of the different possible screws orientations. Once the prediction is made, two clusters are created owing to a K-Means algorithm to separate the cluster of letters (top), and the cluster of numbers (bottom), bounding boxes associated with the Hilti's logo form another cluster [Fig.S2-column 1]. The primary orientation of the top and down clusters is then determined. If the cluster is horizontally orientated [Fig.S2-case 1], the bounding box associated with the minimal x coordinate constitutes the extremity of a linking chain, reciprocally if the cluster vertically orientated

[Fig.S2-case 2,3] the bounding box associated with the minimal y coordinate defines the extremity. From this extremity, the linking chain associates with the last element included its first closest neighbor, not already included. For the two clusters, this process allows generating two lists of bounding boxes ordered according to bounding boxes coordinates. The two resulting lists are ordered to position the elements corresponding to the cluster of letters in front of the elements belonging to the cluster of numbers. Finally, the famous algorithm of Needleman and Wunsch algorithm [24] is used to align the expected label and the predicted one. According to the screw orientation, the linking chains built have to be reversed [Fig.S2-case 3]. Therefore, two alignments are calculated, and the one associated with the best score is kept. The Needleman and Wunsch's algorithm is also used to align the logo; if the reverse solution is retained, then the logo text is also reversed. The described strategy answer to most of the cases but it is not exhaustive, for examples:

- If a full cluster is predicted in reversed order the reversion operation will cancel the potential miss-classifications, *e.g.*, for an expected cluster "KH" if the prediction is "HK" the reversion operation implies that the prediction becomes "KH";
- If a miss-classification is upstream of a FN, the FN position is difficult to determine.

Given the excellent performances of the model, despite these rare exceptions, this strategy has been retained.

2.10 Detection of damaged characters

Object detection algorithms allow the classification and the localization of objects. This property solves the first problem that it is to say they indirectly allow guaranteeing that the right stamp is used according to the expected label given the material number being manufactured. Nevertheless, these methods can hardly be used to catch the quality of the characters engraved. Effectively, we also aim to catch damaged characters resulting from the wear of the stamp, which typically implies partial deletions of the characters. We noticed that the accuracies distributions per character are centered around 99% with small variance. Therefore, the model tends to take a binary decision concerning the bounding boxes' classification.

To overcome this issue, we extended the training set with crop pictures centered around a character labeled and including partially another character not labeled [Fig.S3a][21]. Furthermore, we created artificially damaged pictures, since it is impossible to have enough damaged objects to cover most of the possible cases. These artificial damages have been created by various techniques such as mask in-painting operations [Fig.S3b], partial covering of characters using background matrices [Fig.S3c], or pixels operations on character' edges [Fig.S3d]. The artificially damaged characters have not been labeled since a damage class would have gathered too various objects. Hence, the model has been trained

[21] These pictures will be referred to as the "1.5 cropped pictures"

to ignore the damaged characters. These two techniques allowed defining thresholds of uncertainty for each character; like this, if the accuracy associated with a character is too low, the user can choose to accept or reject the proposal.

2.11 Application: "Hilti Fast Falcon"

The application has been build in several stages:

1. The first version have been implemented to run locally and to guarantee the reading of the screws; [Fig.2-b];
2. The second version solves the reading of the stamps [Fig.2-c];
3. The third version integrated a Multi-clients/Server architecture to boost the inference process's speed owing to the GPU.

The setup strategy allowed rapidly testing the prototype on the production line. The GUI has been developed according to the Model View Controller (MVC) model. The model has been divided into sub-modules allowing the following operation: validation of the material number; picture acquisition; prediction of the label with a commercial OCR if the request refers to a stamp, or with our object detection model if the request refers to a screw; comparison between the detected label to the expected one. The View is the graphical interface that handles the windows' creation, the recording of users' actions, the display of the results, and the display of different alerts. The Controller is at the interface between the View and the Model; it handles the communication between users' requests and the Model. This architecture has been extended with a Multi-client/Server interface, where communications are handled through the TCP/IP protocol. If the connection between the client and the Server is established, the server holds the Model, and all user' requests through the GUI are translated into "client requests" that are submitted to the Server. The interactions between the clients, the Server, and the Model are coped with another interface handling different threads of processes. The resulting architecture will be detailed in the results section.

3 Results

3.1 Comparison of commercial OCR services

Before developing our own "characters recognition" algorithm, we assessed three commercial OCR services performances, since this solution appears to be easy to develop and maintain. Firstly, the performances of Google, Azure, and AWS OCR service [8, 9, 10] have been assessed on an external data set, gathering natural scene pictures containing text written in the Latin alphabet. This experiment allows detecting the eventual biases resulting from our data set [Fig.5-a]. The results obtained in the external data set are

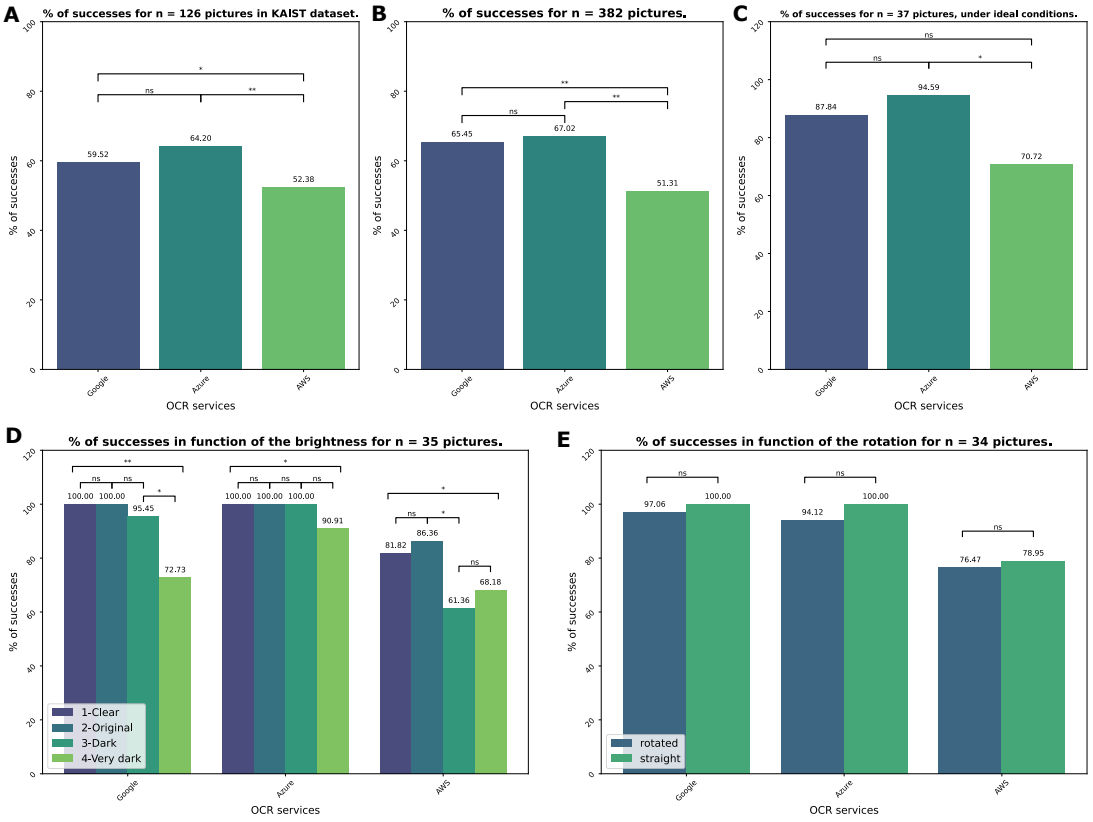


Figure 5: **Comparisons between Google Azure and AWS OCR services [*].** **A)** Percentages of successes of the three OCRs on 126 pictures taken from an external data set: KAIST English scene text database [25]. **B)** Percentages of successes for 382 pictures of stamps [Fig.S1-c], with different cameras, orientation and lighting conditions. **C)** Percentages of success for 37 "ideal" stamps pictures, *i.e.*, the objects have been placed in the center of the picture, straight in comparison to picture's frame, and under optimal lighting condition. **D)** Percentages of successes according to pictures' brightness. To get comparable results, pictures brightness, has been processed adding the value 20 to all pixels of the "Original" pictures to get the "Clear" cases, and removing respectively -30 and -50 to get the "Dark" and "Very dark" pictures. **E)** Assessment of OCRs de-skew technique, photographing the objects with an angle between $\pm[60; 120]$. For each comparison, χ_2 independency tests have been realized, "ns" implies that the hypothesis H0 cannot be rejected, "*" that H0 has been rejected under a risk $\alpha = 0.05$, and "**" that H0 has been rejected under a risk $\alpha = 0.01$.

comparable to those resulting from the analysis of 382 pictures of stamps taken under various lighting conditions, with different object orientations and with different cameras [Fig.S1-c]. Like this, we confirmed the quality of our data set. It appears that Google and Azure OCR services highly outperform AWS OCR technology. Furthermore, if the difference between Google and Azure's performances is not statistically significant, numerically Azure outperforms Google. In a second time, we evaluated the effect of pictures' brightness to determine the lighting requirement. AWS is highly sensitive to brightness modifications, probably implying poor pre-processing techniques, especially binarization methods [Fig.5-d]. Google shows some degradation of its performances when pictures are

"very dark", whereas Azure does not seem to be affected by these modifications [Fig.5-d]. Thirdly, in order to assess the de-skew methods of these services, we evaluated OCRs' performances under two conditions: with the objects arranged parallelly to the pictures' frame ("straight") and with an angle of $\pm[60; 120]$ ("rotated"). Objects' rotation does not affect OCRs' performances [Fig.5-e]. Finally, under optimal conditions near to the ones that can be found in the production line, *i.e.*, with constant lighting and constant distance between the object the camera, and with a reasonable object-orientation according to the pictures' frame, we show that Azure outperforms Google and AWS [Fig.5-c] [22].
The second batch of evaluations has been done for the screw case [Fig.S1-b]. Because of text orientation written in a circle around the screws, any OCR service has been to read successfully one picture. Given the numerous advantages of these services compared to a homemade implementation, we tried to pre-process the images to linearized the text. The best pre-processing method found implies the following steps [Fig.S4-a]:

1. Increase of pictures' contrast, clarity, and brightness, then blur the pictures with a Gaussian filter (These operations allow reducing the noise for the next step.);
2. Find the center of the screw;
3. Apply a log-polar transformation to linearized the circle;
4. Assuming that the objects have a constant orientation, flip at 180° the left side of the linearized picture, to get all the text into the same orientation.

This methodology allowed getting 64% of successes with Azure and respectively 59% and 52% with Google and AWS [Fig.S4-b]. Those disappointing results can be explained by the fact that characters are deformed after the enumerated transformations. Because of the difference between these results and a target accuracy of 99%, this drafted strategy has been dropped.

3.2 Faster R-CNN: Comparison between two features extractor CNNs

As introduced, the literature review led us to focus our study on the Faster R-CNN model since it appears to be the best technique in terms of accuracy. We compared the latest two published feature extractor networks, *i.e.*, Inception-v2 and Inception-v4-Resnet-v2, that we will respectively be called in this section Inception and Resnet. Considering the same data set and the same training time, both models manage to learn the training data since the loss curves converge toward 0 [Fig.6-a]. Both models suffer neither from overfitting nor under-fitting, given the training trends and validation loss curves [Fig.6-a]. This statement can also be made; because each model, there are no statistical differences between the successes' proportions between the training and the validation sets [Tab.2]. Analyzing

[22] The nonsignificance of the statistical tests between Azure and Google can probably be explained by the relatively small size of the data set only 37 pictures.

the main trends of the training process [Fig.6-a-b-c] we noticed that Resnet outperforms Inception. Firstly according to the loss curves, it appears that the training process is faster and more stable with Resnet [Fig.6-a]. Resnet's faster speed of convergence is also clearly visible on the mAP and AR validation curves [Fig.6-b-c]. Secondly, analyzing the mAP curves [Fig.6-c], it emerges that Resnet is more accurate, with at the end of the training, an mAP equivalent to $\simeq 0.82$ against $\simeq 0.78$ for Inception. Thirdly, Resnet is more sensitive, given the analysis of AR curves, which converge toward $\simeq 0.86$ against $\simeq 0.79$ for Inception [Fig.6-c]. These general analyses are confirmed by the comparisons of the proportions of successes between the two models, and especially by the comparison of the proportions of pictures correctly labeled, which is the most critical metric, and which is significantly higher for the three sets with Resnet as features extractor network [Tab.2]. Analyzing, in detail, the results on the test set, we can point out that the two models managed to learn each screw label [Fig.6-d-f]. This demonstrates, on the one hand, the pictures' homogeneity between the test and the training set and, on the other hand, the constant quality of pictures between the different screws. This also shows the model's ability to learn characters of different font sizes since the results are similar between screws of different diameters. Finally, enumerating the different errors on the test set, we can point out the very low number of failures in both cases [Fig.6-e-g]. Especially the excellent performances of Resnet are noticeable knowing that the miss-classification of "4" to "x" results from labeling error and that the false negative "/" can be explained by the presence of a scratch [Fig.6-g]. Given this sum of information, we chose Inception-v4-Resnet-v2 as a feature extractor network for its accuracy and speed.

	Train			**Test**			**Validatition**		
	Inception	Resnet	s	Inception	Resnet	s	Inception	Resnet	s
% pictures correctly produced	$\frac{4992}{5190}$ = 96.185%	$\frac{5188}{5190}$ = 99.942%	*	$\frac{1211}{1224}$ = 98.938%	$\frac{1222}{1224}$ = 99.837%	*	$\frac{339}{348}$ = 97.414%	$\frac{348}{348}$ = 100%	*
% bounding boxes correctly predicted	$\frac{49846}{50040}$ = 99.612%	$\frac{50039}{50040}$ = 99.999%	*	$\frac{11764}{11775}$ = 99.966%	$\frac{11775}{11775}$ = 100%	NS	$\frac{3367}{3375}$ = 99.763%	$\frac{3375}{3375}$ = 100%	*
% characters correctly predicted	$\frac{49769}{50040}$ = 99.458%	$\frac{50039}{50040}$ = 99.999%	*	$\frac{11763}{11775}$ = 99.989%	$\frac{11773}{11775}$ = 99.992%	NS	$\frac{3366}{3375}$ = 99.763%	$\frac{3375}{3375}$ = 100%	*

Table 2: **Summary of the performances of Faster R-CNN with Inception-v2 (referring to Inception) and Inception-v4-Resnet-v2 (referring to Resnet) as features extractor CNNs [*].** For the first row of comparisons, a picture is said to be entirely correctly predicted if all bounding boxes have been found and correctly qualified. The deviation between the proportion of bounding boxes correctly predicted and the proportion of characters correctly predicted comes from the characters successfully located but miss-classified. The "s" columns refer to the statistical significance of the unilateral proportion Z-tests under the null hypothesis (H0) that Resnet outperforms Inception. Like this, the "*" symbols imply that H0 has been rejected under the risk $\alpha = 0.05$, and "NS" that the difference is not significant.

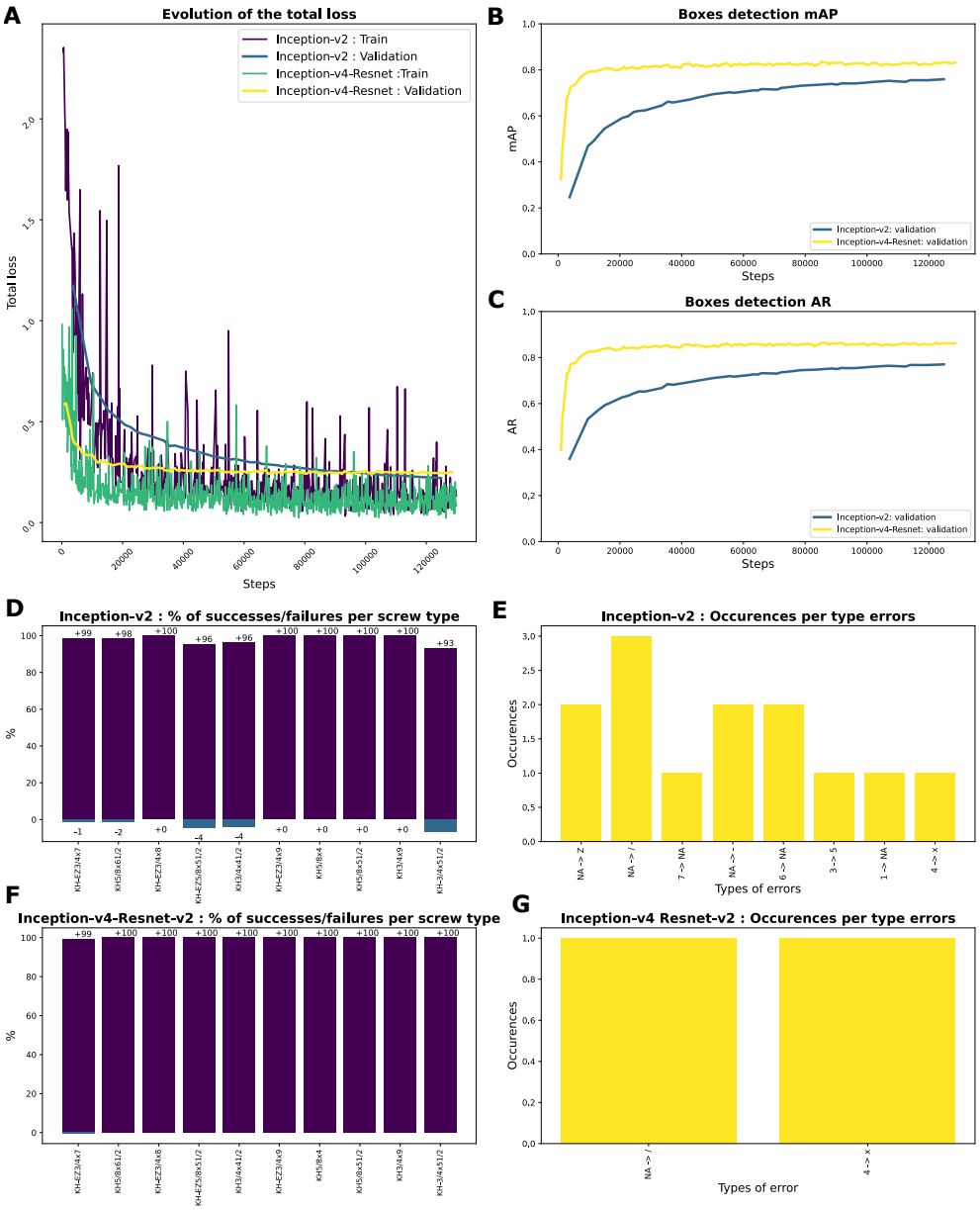


Figure 6: **Comparison of Inception-v2 and Inception-v4-Resnet-v2 as feature extractor CNN [*]. A)** Evolution total loss for the training set and the validation set with Inception-v2 and Inception-v4-Resnet-v2. **B)** Box detection mAP (mean average precision) for the validation sets. The mAP metric is the mean of the areas under the precision/recall curve for a given set of recall levels [26]. For a reminder, the precision is defined as (TP/(TP+FP)), and the recall as (TP/(TP+FN)). **C)** Average recall (AR) for the validation sets. The average recall is the value of integral of the recall over the IoU. (*The recall values are inversely proportional to the IoU*). **D)** Proportions of pictures correctly and incorrectly predicted according to the different types of screws with inception-v2 as a feature extractor. In the User Interface (UI), the screws of diameter 1/2 have been excluded. The high level of errors observed associated with the screw labeled "KH-EZ 5/8 x3 1/2" can be considered an outlier values because of picture quality. **E)** Occurrences of all kinds of errors, with Inception-v2 as a feature extractor. $x \rightarrow$ NA corresponds to a FN, *i.e.*, the character x must have been detected; NA $\rightarrow x$ corresponds to a FP, *i.e.*, x must not have been detected; $x_1 \rightarrow x_2$ corresponds to a miss-classification of the character x_2 that must have been detected as x_1. **F)** and **G)** are identical to **(D)** and respectively **(E)** with Inception-v4-Resnet-v2 as a feature extractor CNN.

3.3 Detection of damaged characters

The increasing size of the training set with "1.5 cropped pictures" [Fig.S3a] and various damaged pictures allowed widening accuracy distributions. For example, for the characters "K", "-" and "1" these experiments decreased the mean accuracy by respectively 0.17 (p-value = 0.12[23]), 0.77 (p-value = 2×10^{-6}), and 3.96 (p-value = 5×10^{-9}) and so increased the distributions' variance. Also, the network learned to ignore all the damaged characters [Fig.S3b, S3c, S3d].

3.4 Application: "Hilti Fast Falcon"

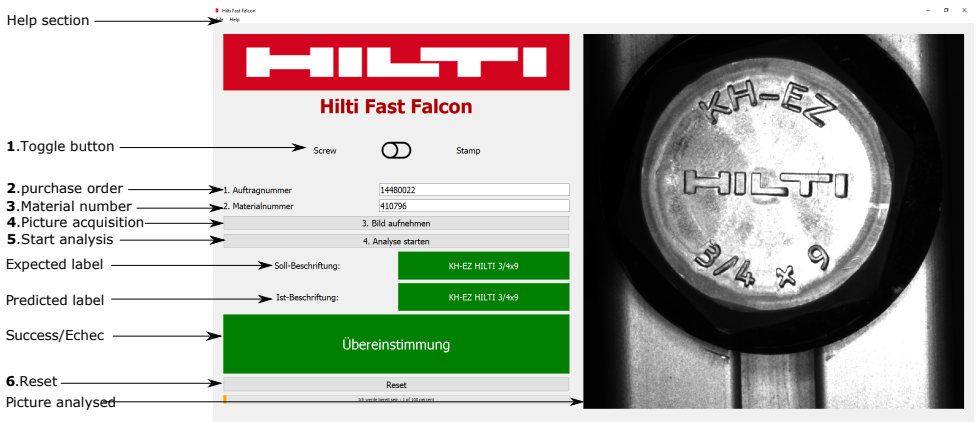
The application "Hilti Fast Falcon" is the final "product" of our case study. It has been implemented to work on the production line computers that are plugged to cameras and connected remotely to a server, which is an eGPU. When the application is loaded, users have first to select the type of analysis, *i.e.*, screw or stamp analysis [Fig.7a-1]. Then, he/she scans the bar code associated with the purchase order being produced [Fig.7a-2]. If this one is readable, it automatically fills the material number field [Fig.7a-3]. The expected label associated with this material number is then searched into the Eilebrech database. The expected label is returned if it corresponds to a material number associated with a screw of diameter 3/4 or 5/8 inches, or anchor with a linear label [Fig.S7-Grey boxes]. Once the object is correctly lighting and placed under the lens, the user takes a picture [Fig.7a-4]. If the camera is not recognized, a picture locally stored on the computer can be loaded. The picture to analysis is displayed on the GUI as well as the expected label. If the client socket has been opened this information are sent to the Server [Fig.8-step 13].

If the analysis concerns a stamp picture, this one is loaded into the Azure OCR API [9]), the text predicted is then stored into the queue of analysis to post-process [Fig.S6c-S6b]. The post-processing protocol includes the comparison between the expected label and the predicted one; if it is a match, the result returned is only a Boolean such as `success = True`. Otherwise, the predicted label is aligned against the expected one through the Needleman and Wunsch's algorithm [24]. The results returned are a Boolean turned to false, and the HTML code corresponding to the alignment where the miss-classification are indicating by red letters and the FN by underscores [Fig.8-second Alt frame ; Fig.S6c]. The post-processed results are finally added to a queue of results to send to the clients. After receiving the results, the client communicates with the interface, asking the GUI to display the prediction results.

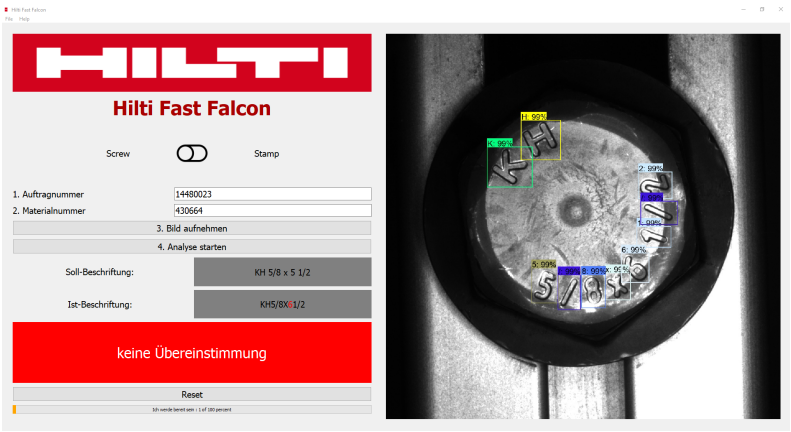
If the request concerns a screw, then it is handled by the Tensorflow model. At this stage, we assume that the model has been initialized, *i.e.*, the Tensorflow session has been opened, and the frozen graph has been loaded. If it is not the case, only the stamps analysis program is available. For screw analysis, the bounding boxes' accuracy

[23] P-values calculated through unilateral the Mann-Whitney rank tests.

is inspected, *i.e.*, if the accuracy associated with a character is too low, the coordinates of this object are stored into a dictionary. However, if any potential error has been detected, the results are directly sent to the queue of analysis to post-process. Otherwise, intermediate steps between the prediction and the post-processing pipeline involving the user are needed [Fig.8-first Alt frame ; Fig.S6a]. For each potential wrong bounding box, the user has to accept or reject the proposal [Fig.8-frame loop]. Like this, the user's opinions are translated into a vector of bounding boxes to delete, which is then sent to the Server. After all, the analysis is ready to be post-processed, once the bounding boxes rejected have been deleted. The post-processing pipeline of screw pictures has a step upstream to the text comparison, which is the organization of the bounding boxes [Fig.S2]. The following post-processing steps are identical. Once the post-processing is done, the results are returned and then displayed on the GUI; if the analysis did not lead to match the labeled pictured, showing the bounding boxes is presented [Fig.7b].From a programming point of view, each object is associated with a pointer toward the client's socket to connect several clients simultaneously. **This architecture allows guarantying an answer in 3 seconds** in the best case,*i.e.*, a match between the expected and the predicted label [Fig.7a ; S6b]. **This is an important improvement comparing to the previous version of the application that was running locally, since using the production line computers one analysis took around 24 seconds.** It also eases the stamp analysis since computers on the production line do not have access to the internet [Fig.S6b ; S6c]. Finally, the architecture set up optimizes access to the GPU computational resources. Since the multi-threading allows predicting a new picture while another analysis is being post-processed, the results of another are being sent to a client, and the server is accepting a new request. It is worth noting that if the server is not functional all the processes happen locally. For more details on the architecture Model-Controller-View Multi-clients/Server, the UML diagram in annex can be referred [Fig.S7].



(a)



(b)

Figure 7: **Global views of the application [*]. A)** Presentation of the different functionalities and description of a screw's picture analysis that led to a match between the expected and the predicted label. **B)** Screw's picture analysis that led to a mismatch, in this case, the labeled picture and the best alignment of the predicted label resulting from the Needleman and Wunsch are shown.

4 Discussion

4.1 Improving the model

4.1.1 Refine damaged characters detection

As shown by the results from [Tab.2], the current Faster R-CNN model almost perfectly predicts the engraved characters. Nevertheless, as introduced previously, this model hardly makes the distinction between good and poor quality engravings.Despite the experiments done to increase accuracies distributions' variance and to learn to ignore damaged characters, this set up shows several limits. Firstly, accuracy values reflect the probability that a given character belongs to the class k knowing the probabilities that this character belongs to the other classes. Therefore even if this probability should be lower for imperfect characters, it does not reflect properly speaking characters' quality. Secondly, although the results on artificially damaged characters are encouraging, since the network perfectly learned to ignore them, we cannot infer from these results that the network will have the same behavior on real damaged characters. To confirm this assumption, we need a single real example; we are currently working on damaging stamps. To bring a better answer to this issue, a continuous variable correlated to the characters' quality is needed; that is why we are shifting our attention toward the Mask R-CNN model, which is an extension of Faster R-CNN [27]. In addition to predicting objects' class and localization, Mask R-CNN returns a semantic segmentation of each object detected. It predicts objects' mask, determining which pixels belong to this instance. Segmentation masks are inferred for each RoI in parallel with the branch for classification, and with the branch for bounding boxes regression [27].

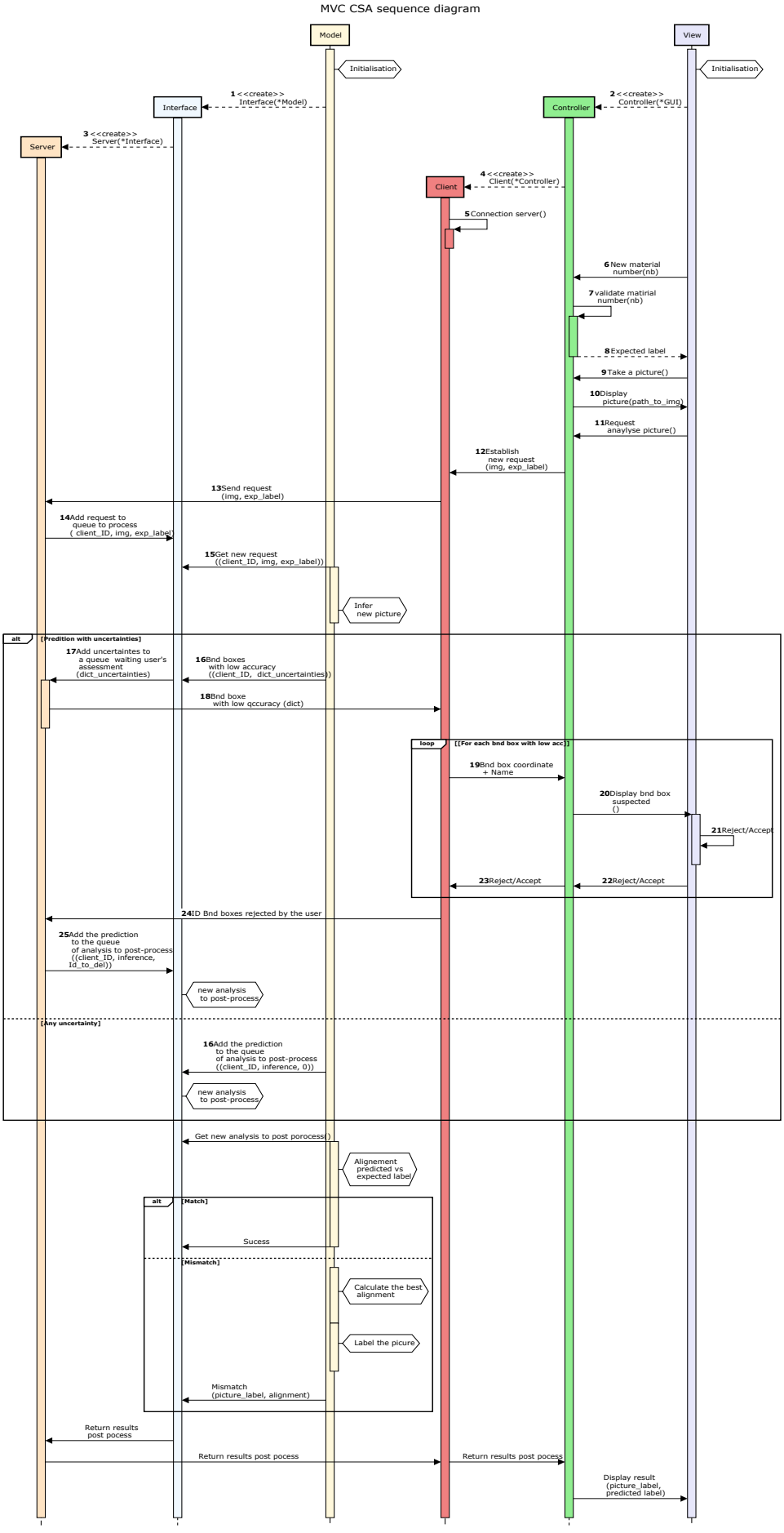

MVC CSA sequence diagram
Model
View
Initialisation
Initialisation
Interface
Controller
1 <<create>> Interface(*Model)
2 <<create>> Controller(*GUI)
Server
3 <<create>> Server(*Interface)
Client
4 <<create>> Client(*Controller)
5 Connection server()
6 New material number(nb)
7 validate matirial number(nb)
8 Expected label
9 Take a picture()
10 Display picture(path_to_img)
11 Request anaylyse picture()
12 Establish new request (img, exp_label)
13 Send request (img, exp_label)
14 Add request to queue to process (client_ID, img, exp_label)
15 Get new request ((client_ID, img, exp_label))
Infer new picture
alt
[Predition with uncertainties]
17 Add uncertaintes to a queue waiting user's assessment (dict_uncertainties)
16 Bnd boxes with low accuracy ((client_ID, dict_uncertainties))
18 Bnd boxe with low qccuracy (dict)
loop
[For each bnd box with low acc]
19 Bnd box coordinate + Name
20 Display bnd box suspected ()
21 Reject/Accept
23 Reject/Accept
22 Reject/Accept
24 ID Bnd boxes rejected by the user
25 Add the prediction to the queue of analysis to post-process ((client_ID, inference, Id_to_del))
new analysis to post-process
[Any uncertainty]
16 Add the prediction to the queue of analysis to post-process ((client_ID, inference, 0))
new analysis to post-process
Get new analysis to post porocess()
Alignement predicted vs expected label
alt
[Match]
Sucess
[Mismatch]
Calculate the best alignment
Label the picure
Mismatch (picture_label, alignment)
Return results post pocess
Return results post pocess
Return results post pocess
Display result (picture_label, predicted label)

Figure 8

Figure 8: **Sequence diagram e of screw picture analysis according to the MVC Multi-client/Server architecture [*].** This diagram summarizes the interactions between each module. The GUI (violet boxes) handle by the users modifies the state of the Controller (Green boxes), which is linked to the Client (Red boxes), if the Server (Oranges boxes) is connected, allowing remote management of the requests. The requests sent to the Server are shared with the Interface (Blue boxes). The Interface copes with requests' order and sends them to the Model (Yellow boxes). Firstly, the Model handles picture prediction. If the prediction contains potential mistakes, the bounding boxes associated with low accuracy are sent to the user, that have for each of them have to accept or reject the proposal (alt and loop). The IDs of bounding boxes rejected are then sent back to the Server that sends a request to the interface to post-process the current analysis. The post-processed results are finally displayed on the GUI.If any bounding box is considered a potential error, the prediction is directly post-processed before that the final results are displayed on the GUI.

Considering a model with k classes, for each RoI, the mask branch has a dimensional output equivalent to km^2, encoding k masks of resolution $m \times m$. The multi-task loss [Eq.1], therefore becomes $L = L_{class} + L_{loc} + L_{mask}$, where L_{mask} is defined as the average binary-cross entropy loss. It only includes k^{th} mask if the region is associated with the ground truth class k, which is calculated such as

$$L_{mask} = \frac{-1}{m^2} \sum_{1 \leq i,j \leq m} y_{i,j} log(\hat{y}_{ij}^k) + (1 - y_{i,j}) log(1 - \hat{y}_{i,j}^k).$$

In this equation $y_{i,j}$ refers to the label of the cell (i, j) in the true mask inside the region of size $m \times m$, and $\hat{y}_{ij}^k$ is the predicted label for this cell, by the mask inferred for the ground class k. This definition allows decoupling mask and class prediction since a mask is created for every class and the selected mask results for the class label predicted with the classification branch. The most crucial difference between Faster R-CNN and Mask R-CNN is the replacement of the RoI pooling layer by a RoI align operation. We can remind that the pooling RoI layer allows the extraction of small feature maps from each RoI. This operation successively applies two quantization operations. Firstly, map the anchor into the feature map and, secondly, apply the pooling filter These operations led to a loss of precision, and then so data losses. Since the instance segmentation requires a much finer spatial representation of the objects, the RoI align layer properly aligns the extracted features. Computing the exact coordinates of the input features, using four regularly sampled locations in each RoI bin, and aggregating them using a max average pooling layer. Technically, the shift from Faster R-CNN to Mask R-CNN will be time-consuming since the train set as to be re-labeled to create masks. .We will use the open-source program Pixel Annotation Tool[24], which proposes a semi-automatic annotation due to an edges recognition algorithm. Furthermore, we hope that our semi-supervised labeling

[24] https://github.com/abreheret/PixelAnnotationTool

protocol could be reusable, even though highly accurate masks should be created. We will use the implementation proposed on the official object detection Tensorflow repository [25], re-using the weights gotten with Faster R-CNN as a transfer learning process. This model appears to be promising since it will allow comparing the masked area predicted to the expected one. This would allow detecting easily damaged characters, assuming that we will be able to create highly accurate masks for a set of good quality characters that would be used to define the reference areas per character.

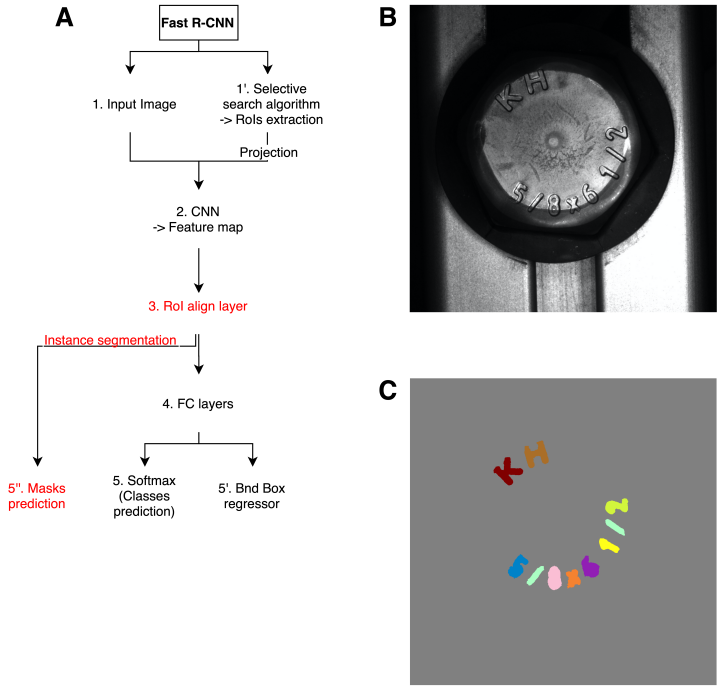


Figure 9: **From Faster R-CNN to Mask RCNN [*]. A)** Mask RCNN architecture: the steps in red represent the Mask R-CNN specificity in comparison to Faster R-CNN. **B)** The picture to predict. **C)** Mask prediction using Inception-V4-Resnet-V2. The different colors represent the masks per character. As observed in this picture, currently, the areas detected are not accurate enough to be compared to the reference areas. Especially we can point out that the network did not learn the inner circles from the closed characters, *i.e.*, 8, 6, and 5.

4.1.2 Refine the alignment between the predicted and expected labels

Another improvement of the model to bring is the writing of the predicted label. The solution currently implemented allows in all cases stating with certainty if the prediction match with the expected label. Nevertheless, the alignment returned in the case of a miss-match is not always relevant. For example, FN position should respect the expected localization of the missing bounding box and not be positioned to maximize Needleman and Wunsch's score, like this aligning the expected label $KH-EZ\ HILTI\ 5/8X6\ 1/2$ with a screw labeled $KH-EZ\ HILTI\ 3/4X9$ we get $KH-EZ\ HILTI\ 3/4X\_\_9$. In

[25] https://github.com/tensorflow/models/blob/master/research/object_detection/samples/configs/mask_rcnn_inception_resnet_v2_atrous_coco.config

contrast, we would like $KH{-}EZ\ HILTI\ 3/4X9\_\_$. Therefore, we aim to implement a solution focused on characters spatial localization.

4.2 Improving the application

Users are globally satisfied with the application; it is intuitive and robust. Especially, coaches and quality managers are contented with its accuracy. The main drawback of "Hilti Fast Falcon" is its speed since around 24 seconds are needed to get the result of one analysis. This inconvenience has been solved by implementing the Multi-clients/Server architecture. With the GPU used for training, the network takes only 3 seconds to predict a screw label. We also experimented a `C++` implementation of the inference stage without any significant speed improvement; therefore, regarding the complexity of this implementation, we finally kept the Python one. We are still realizing numerous tests to create a server as robust as possible. Effectively, the server will need to be able to handle till 17 clients and run 24/7. Also, the manageability of the Multi-client/Server architecture will be another essential advantage since all the future releases could be mostly done from the server.
In the next releases, we will use the images correctly predicted to increase the size of the training set and to program a new training session periodically. This technique should help us to get more robust predictions along the time.

5 Conclusion

The application developed allows a preferment prediction of the curved and linear text embossed on steel surfaces. First, it allows detecting relevantly the linear text engraved on stamps owing to the Azure OCR service, which has been shown as the best OCR software in comparison to Google and AWS services. Secondly, It allows highly accurate detection of the circular labels punched on screws. Therefore, we showed that Faster R-CNN is a high-performance object detection technique that can be used for quality control tasks requiring an ultra-precision. The implemented model solves the challenges of reading characters, whatever their orientation, and their size in various lighting conditions. This model has been integrated into an intuitive GUI, whose Multi-client/Server architecture guarantees an answer in an optimal amount of time on a classic computer connected remotely to a GPU. Shortly, the application will be extended to more products, primarily to anchors whose label is written at 360° around the body of the tool. Finally, the issue to solve concerns the early detection of the wear of the stamps. That is why we are implementing a Mask R-CNN model that will allow comparing the expected area of the characters to the detected one.

References

[1] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[2] Honggang Zhang, Kaili Zhao, Yi-Zhe Song, and Jun Guo. Text extraction from natural scene image: A survey. *Neurocomputing*, 122:310–323, 2013.

[3] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[7] Zongyi Liu and Sudeep Sarkar. Robust outdoor text detection using text intensity and shape features. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[8] Google: Detect text in images. https://cloud.google.com/vision/docs/ocr. Accessed: 2020-08.

[9] Azure: Optical character recognition (ocr). https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-recognizing-text. Accessed: 2020-08.

[10] Aws: Detecting text. https://docs.aws.amazon.com/rekognition/latest/dg/text-detection.html. Accessed: 2020-08.

[11] Gerhard Linß. *Qualitätsmanagement für Ingenieure*. Carl Hanser Verlag GmbH Co KG, 2018.

[12] Sunita Parashar and Sharuti Sogi. Finding skewness and deskewing scanned document. *Department of Computer Sc. & Engineering, HCTM, Kaithal, Haryana (India), IJCTA*, 2012.

[13] Umberto Michelucci. *Advanced applied deep learning: convolutional neural networks and object detection*. Springer, 2019.

[14] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[16] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al.

Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

[22] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[24] Cold Spr. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Mol. Biol*, 48:443–153, 1970.

[25] Kaist scene text database. http://www.iapr-tc11.org/mediawiki/index.php/KAIST_Scene_Text_Database. Accessed: 2020-08.

[26] Mark Everingham and John Winn. The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 8, 2011.

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[*] Personal figure.

Acronyms

AI Artificial Intelligence. 1

AR Average Recall. 16, 22, 23

BIM Building information and modeling. 1

BU Business Unit. 1, 2

CNN Convolutional Neural Network. 4, 7–9

FC Fully connected layer. 8, 12

FN False Negative. 17, 18, 23, 24, 29

FP False Positive. 16, 17, 23

GPU Graphics Processing Unit. 15, 16, 19

GUI Graphical User Interface. 4, 16, 19, 24, 25, 28, 30, 39

IoU Intersection over Union. 11, 12, 16, 23

mAP mean Average Precision. 16, 22, 23

MVC Model View Controller. 19, 39

NMS Non-Maximal Suppression. 12

OCR Optical Character Recognition. 3–7, 20, 30, 37

R-CNN Region-based CNN. 2, 3, 7–9, 14–16, 21, 26, 28–30

RoI Region of Interest. 7, 8, 26, 28

RPN Region Proposal Network. 11

SGD stochastic Gradient Descent. 13

SSD Single-shot detector. 8

TP True Positive. 16, 17, 23

UI User Interface. 23

YOLO You only look once. 8

6 Supplementary figures

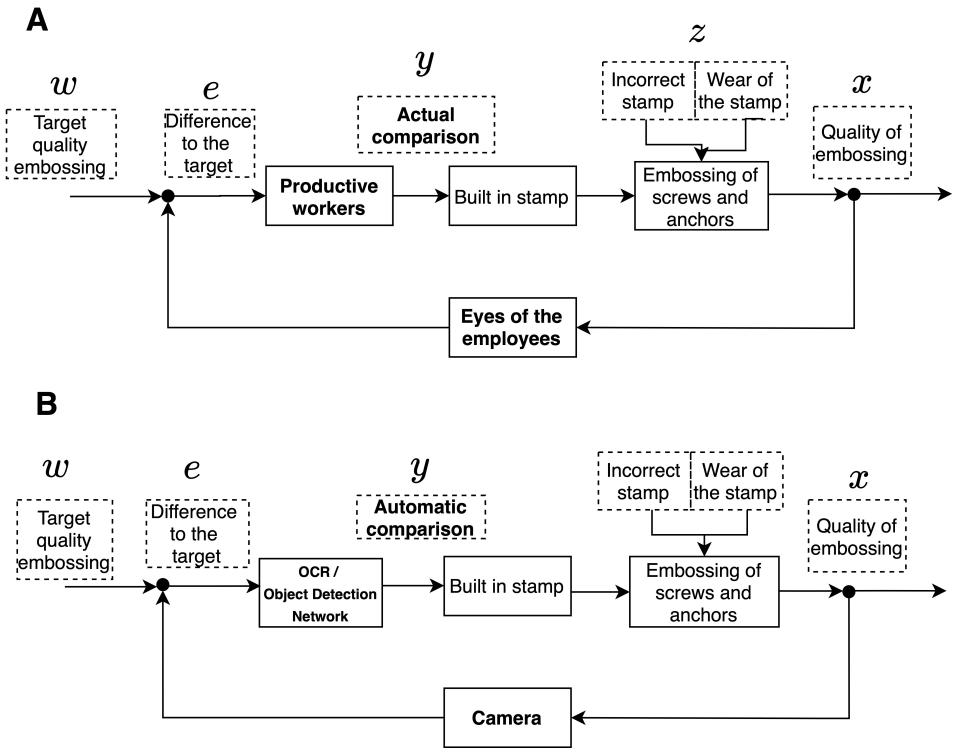


Figure S1: **Quality control cycles [*][11]. A)** Current quality control cycle based on visual control. **B)** Quality control cycle in implementation grounded on computer vision techniques.

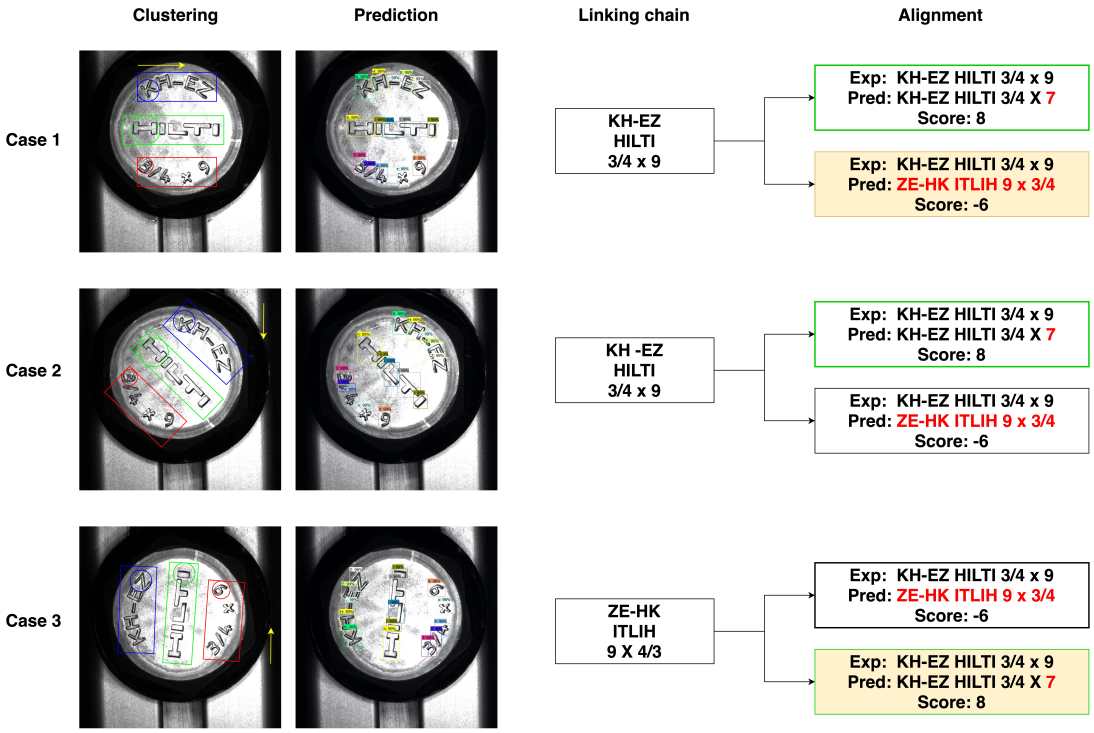
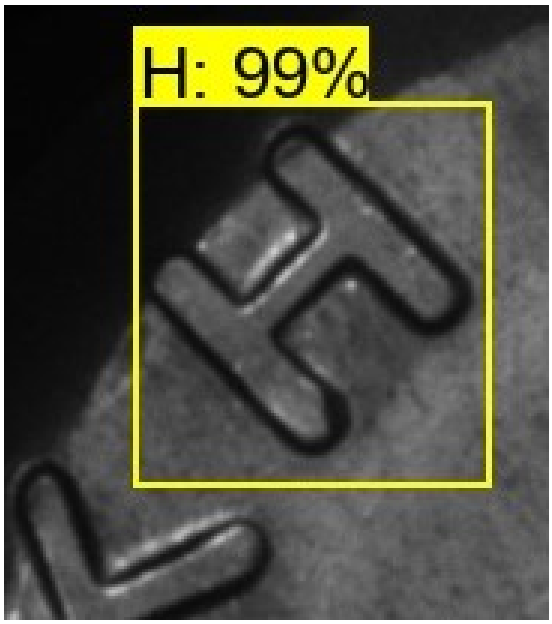
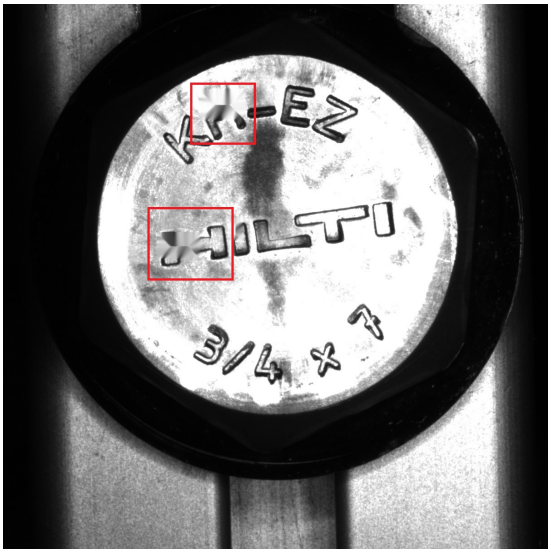


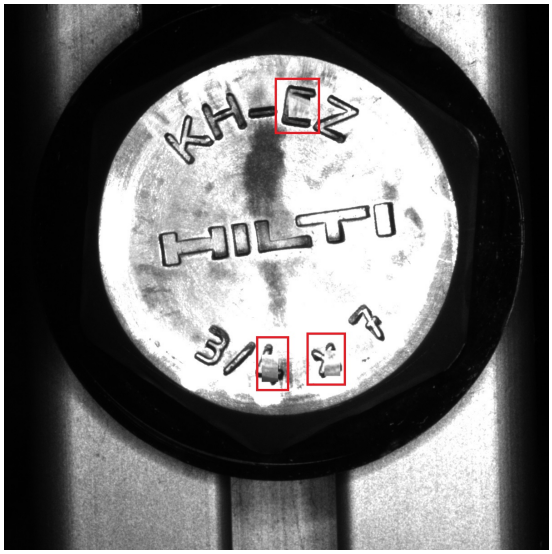
Figure S2: **Comparison between the expected and predicted text [*].** The first column presents the tree clusters with in blue the cluster of letters, in red the cluster of numbers and in green the logo's cluster. In the first case, they are horizontally orientated, and in the following cases are vertically orientated. The characters circled define the linking chain extremity, and the yellow arrow the reading direction expected. The second column presents the bounding boxes predicted. The third one the three linking chain. The last column present the two alignments resulting from Needleman and Wunsch's algorithm [24], with yellow background the one resulting from the reversion of the linking chains. Finally, The best alignment retained is circled in green.



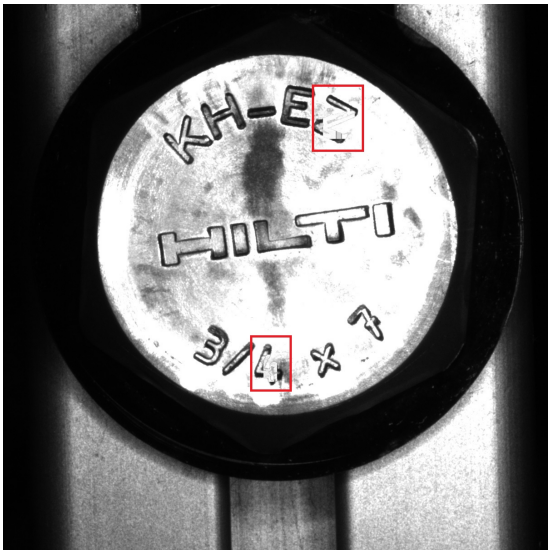
(a)



(b)



(c)



(d)

Figure S3: **Artificial extension of the training set [*]. A)** Crop picture including one character labeled and partially another one unlabeled (1.5 crop experiment). **B)** Inpainting operations to damaged the characters to damage. **C)** Background sub-matrices displacements covering partially the characters. **C)** Pixels operations on characters' edges. On the images B, C, and D the damaged characters are surrounded by a red rectangle.

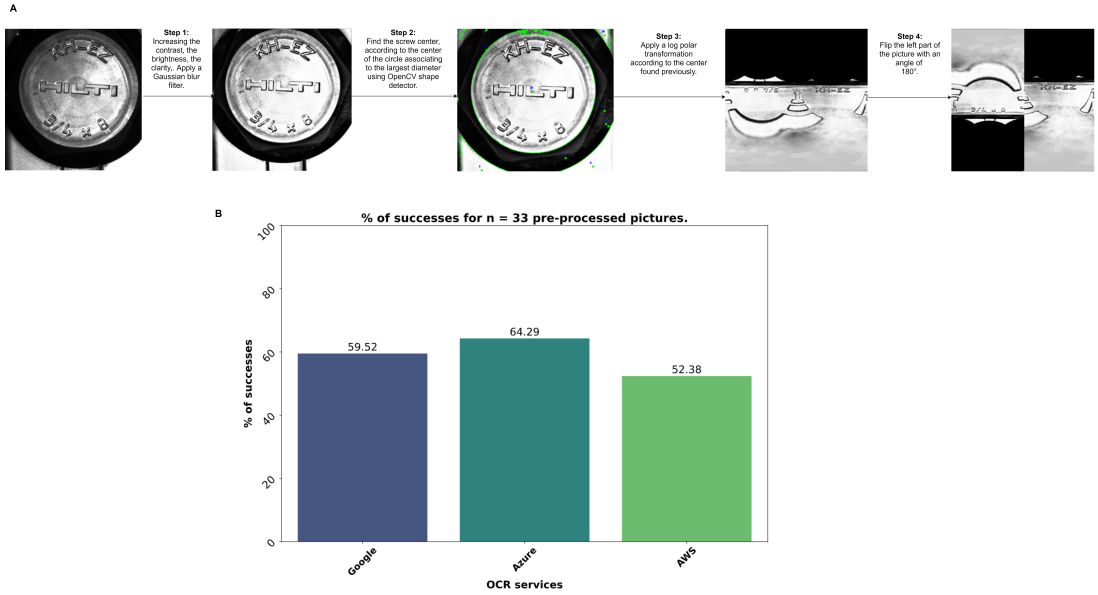


Figure S4: **Assessment of OCR services performances on screw pre-processed pictures. A)** Pre-processing pipeline. **B)** Percentages of successes for each OCR assessed on 33 pictures pre-processed according to the pipeline described.

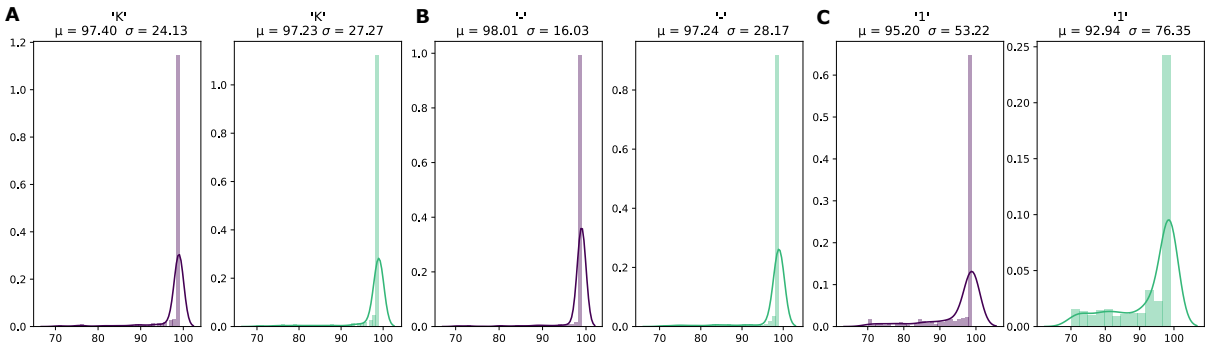
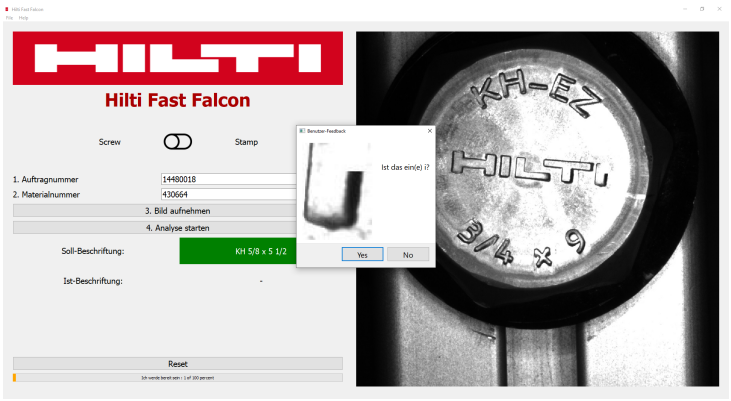
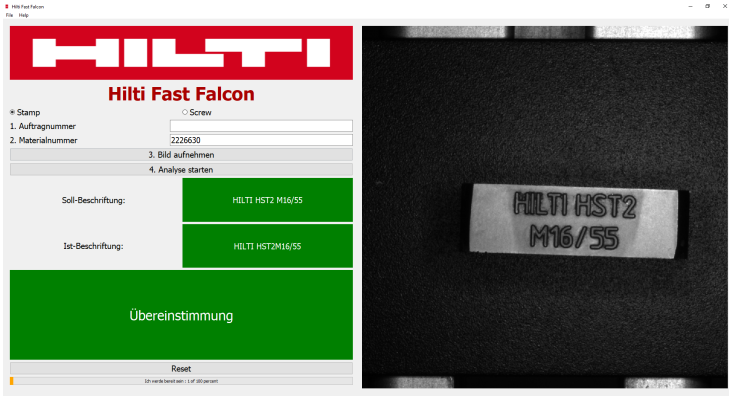


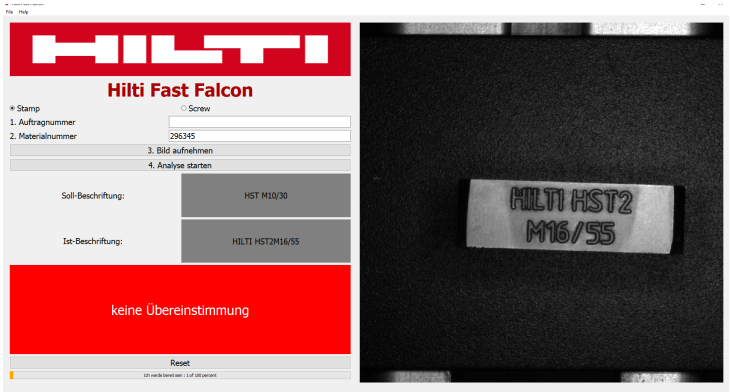
Figure S5: **Effects of the artificial extensions of the training set on accuracies distributions [*].** The accuracies distribution of the character "K" **(A)**, "-" **(B)** and "1" **(C)** have been inferred from the test set the bounding boxes' accuracy. The charts on the left, represent the accuracies distribution before the artificial extension of the training set, like this the charts on the right represent the results of this experiment. μ corresponds to the mean accuracy, and σ the standard deviation.



(a)



(b)



(c)

Figure S6: **Global views of the application [*].** Accuracy alert, linked with the uncertainty of the algorithm concerning the character shown in the pop-up windows. The user has to accept or reject the proposals before the results' are post-processed. **B-C)** Analysis of a stamp picture that led respectively to a match **(B)** and a miss-match **(C)**.

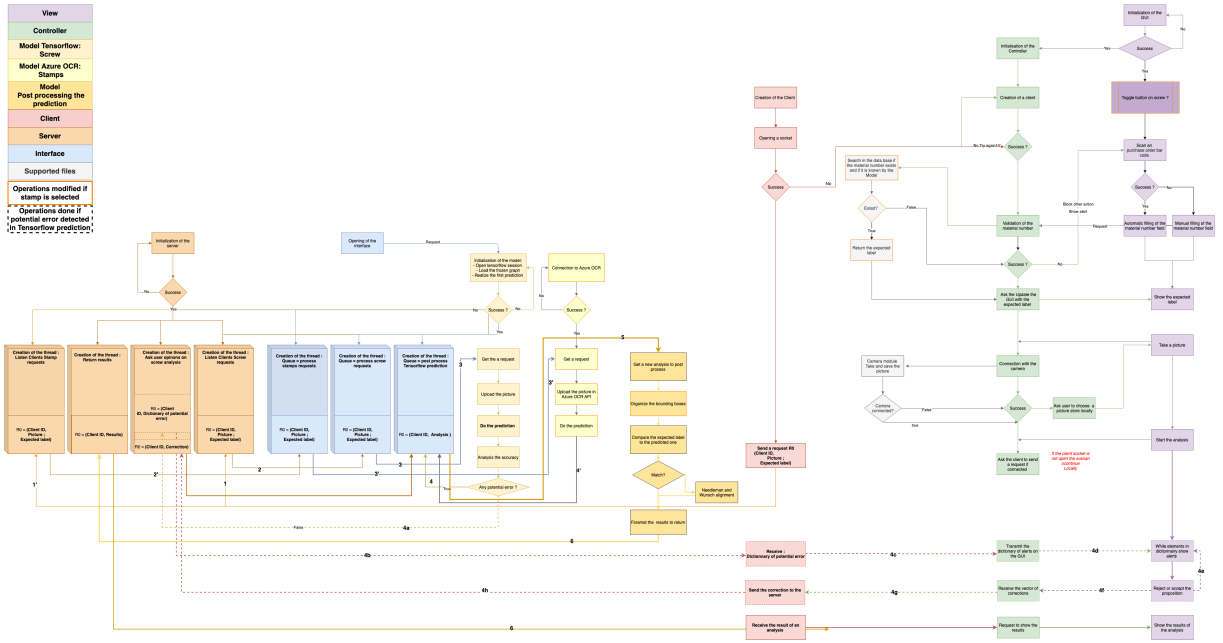


Figure S7: **Detailed of the architecture MVC Multi-client/Server [*].** This diagram summarizes the interaction between the GUI (violet boxes), the Controller (Green boxes) which is linked to the client (Red boxes) if the Server is connected. User's requests are handled by the controller that through the client sends them to the Server (Brown boxes), which is linked to the Interface (blue boxes). The interface copes with requests' order and send them to the Model (Yellow-Orange boxes). The model is divided in three part one handles the request linked to the screw (Clear orange boxes), the other one the request corresponding to stamps (Clear yellow boxes). The results of the predictions are them post-processed (Dark yellow boxes), after to have been corrected by the user (doted line). To get a clearer view of this diagram please follow this link: https://drive.google.com/file/d/1eJXbqYI5QIg-qIZLmVVT8fZpzbt9I5UG/view?usp=sharing

7 Faster R-CNN Inception-v2-Resnet-v4 configuration

```
model {
 faster_rcnn {
 num_classes: 20
 image_resizer {
  keep_aspect_ratio_resizer {
  min_dimension: 600
  max_dimension: 600
  }
 }
 feature_extractor {
  type: 'faster_rcnn_inception_resnet_v2'
  first_stage_features_stride: 8
 }
 first_stage_anchor_generator {
  grid_anchor_generator {
  scales: [0.25, 0.5, 1.0, 2.0]
  aspect_ratios: [0.5, 1.0, 2.0]
  height_stride: 8
  width_stride: 8
  }
 }
 first_stage_atrous_rate: 2
 first_stage_box_predictor_conv_hyperparams {
  op: CONV
  regularizer {
  l2_regularizer {
   weight: 0.0
  }
  }
  initializer {
  truncated_normal_initializer {
  stddev: 0.01
  }
  }
 }
 first_stage_nms_score_threshold: 0.0
 first_stage_nms_iou_threshold: 0.7
 first_stage_max_proposals: 300
 first_stage_localization_loss_weight: 2.0
 first_stage_objectness_loss_weight: 1.0
 initial_crop_size: 17
 maxpool_kernel_size: 1
 maxpool_stride: 1
 second_stage_box_predictor {
  mask_rcnn_box_predictor {
  use_dropout: false
  dropout_keep_probability: 1.0
  fc_hyperparams {
   op: FC
   regularizer {
   l2_regularizer {
    weight: 0.0
   }
   }
   initializer {
   variance_scaling_initializer {
    factor: 1.0
    uniform: true
    mode: FAN_AVG
   }
   }
  }
  }
 }
 second_stage_post_processing {
  batch_non_max_suppression {
  score_threshold: 0.0
   iou_threshold: 0.6
   max_detections_per_class: 100
   max_total_detections: 100
   }
   score_converter: SOFTMAX
  }
  second_stage_localization_loss_weight: 2.0
  second_stage_classification_loss_weight: 1.0
  }
}
train_config: {
 batch_size: 1
 batch_queue_capacity: 10
 num_batch_queue_threads: 10
 prefetch_queue_capacity:5
 optimizer {
 momentum_optimizer: {
  learning_rate: {
  manual_step_learning_rate {
   initial_learning_rate: 0.0003
   schedule {
   step: 900000
   learning_rate: .00003
   }
   schedule {
   step: 1200000
   learning_rate: .000003
   }
  }
  }
  momentum_optimizer_value: 0.9
 }
 use_moving_average: false
 }
 gradient_clipping_by_norm: 10.0
 fine_tune_checkpoint: "PATH/TO/THE/LAST/FROZEN-GRAPH"
 from_detection_checkpoint: true
 num_steps: 200000
 data_augmentation_options {
 random_horizontal_flip {
 }
 }
}
train_input_reader: {
 tf_record_input_reader {
 input_path: "PATH/TO/TRAIN.RECORD/FILE"
 }
 label_map_path: "PATH/TO/LABELMAP.pbtxt"
 queue_capacity: 2
 min_after_dequeue: 1
 num_readers: 1
}
eval_config: {
 metrics_set: "coco_detection_metrics"
 num_examples: 33
 num_visualizations: 10
}
eval_input_reader: {
 tf_record_input_reader {
 input_path: "PATH/TO/VAL.RECORD/FILE"
 }
 label_map_path: "PATH/TO/LABELMAP.pbtxt"
 shuffle: false
 num_readers: 1
 queue_capacity: 1
}
```