Guided-Crop Image Augmentation for Small Defect Classification

Joan Orti, Francesc Moreno-Noguer, Vicenç Puig
Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain

***Abstract*—As processing power becomes more affordable, computer vision tends to push towards controlling complex processes in the industry. Surface inspection, with changing environmental conditions and the usual lack of homogeneity of the inspected parts, makes it a real challenge to overcome even for a skilled specialist. In addition, the scarcity of positive samples and the extremely small size of the defects, makes it even harder to cluster them in different classes. In this work, we propose a novel training strategy tailored to handle these challenges for the problem of image defect segmentation and classification. First, we propose a Context Aggregation Network with different dilation factors, in order to keep as much information as possible from every feature map, especially for the smallest defects. By splitting the loss in classification and segmentation and positively weighing both terms, we accomplish an optimal learning process counteracting possible imbalances in the dataset. Additionally, we introduce a novel guided-crop image augmentation method, which generates new images by cropping real defects from existing images, pasting them in real non-defective ones and finally tweaking their configuration. This augmentation strategically performed, guided by the evolution of each class loss, allows the model to identify better the least common and complicated to identify defects. We validate our solution with the Magnetic Tile and the Severstal Steel Defect Detection dataset, demonstrating that our approach consistently outperforms models such as ResNet-50, DenseNet-121, HRNet or UPerNet.**

I. Introduction

Computer vision has played traditionally a key role in industrial processes as a reliable quality inspection system, ensuring greater repetitiveness and speed than manual inspection. Nowadays, as processing power got more affordable, new horizons are being explored in this field thanks to deep learning [1], [2], [3]. The lack of repetitiveness in a process is not an issue anymore, but instead a bold challenge to overcome. Thus, specialized inspection processes so far only reachable for a human eye, are susceptible to be automated. A fair example would be the surface defect inspection problem, which may feature homogeneous and non-homogeneous surfaces, as well as large and small defects. Precisely the latter, often not bigger than a couple of tenths of millimeter, are the ones that should not be neglected, as they are a potential jeopardy to the integrity of the surface. Consequently, the operator in charge of the inspection must posses a huge experience and skill detecting all sorts of issues in the sheets, even if they are in a non-even surface. That is why the chances of a possible and critical oversight increase, as long as the job is being performed by a human.

In order to automate the process with computer vision, cameras are placed in the line or station in strategic points.

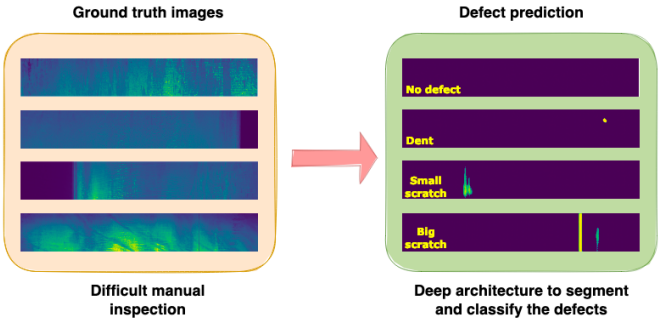

Fig. 1. Examples of line scan camera images of a steel sheet inspection process, featuring different kinds of small defects, like dents or scratches (left). In our work, we propose a deep network to segment and classify these defects in a much faster and reliable way (right).

However, due to the presence of dirt and light changes in the environment, the already challenging task of identifying anomalies in uneven surfaces becomes harder. Even more with the scarcity of positive samples within the images collected by the cameras. Taking steel sheets inspection as a suitable example, several studies featuring deep networks have addressed these problems [4], [5], [6], [7], [8]. Approaching the problem by segmenting and classifying the defects, either they struggle to generalize the model to all classes of anomalies or use a complex augmentation method to deal with the imbalances in the dataset. Therefore, they lack a robust solution that encompasses all classes of defects, keeping at the same time a low complexity to potentially apply it to in different industrial applications. Figure 1 shows an illustration of the desired output of the model, where the images are classified and then segmented, highlighting the corresponding issue.

In this work, we address a multi-class classification and segmentation problem, building upon three main ingredients. First, we propose a Context Aggregation Network [9] with skip connections between opposite dilation layers, to keep the most information from every feature map and thus, helping to identify small defects. We introduced an average pooling layer at the beginning of the network, to reduce computation time without excessively the resolution of the images. We also propose an initial average pooling to speed up the learning process in terms of frames per second (FPS), as well as an interpolation in the final layer to resize the output to the original images size. Second, we designed a composed loss function with two separated terms, classification and segmentation loss, which are balanced during the learning stage to ensure an effective and smooth training. In addition, we introduced a positive weight to both losses, designed specifically to

counteract highly imbalanced class distributions. And finally, we implemented a guided-crop image augmentation technique tailored to the evolution of the training process. The classes with the highest individual loss values are selected to perform this augmentation every certain amount of epochs, cropping parts of real defects, randomly pasting them into a non-defective image and eventually tweaking it with random displacement, rotation and color jitter. That way, we push towards generalizing the defects to every possible background, size and configuration. Although the performance accomplished in this work is pretty remarkable, this augmentation technique can potentially be used on any model.

Our approach is evaluated on publicly available datasets Steel Defect Detection from Severstal [10] and Magnetic Tile dataset acquired by Huang et al. [11]. Both are challenging datasets oriented to surface defect identification in industry applications. The result is a lightweight model, able to get close to 100% mean average precision (mAP) in classification, as well as in terms of mean intersection over union (mIoU) in segmentation, at a remarkable frame rate of 250 FPS.

II. Related work

A. Steel defect detection

Steel defect detection is a defiant surface inspection problem to address in the industry, therefore a good cornerstone to develop a robust identification solution. This problem has two main sides: Speed over accuracy and accuracy over speed. In the work from Fu et al. [3], an image classification is performed using SqueezeNet [12], a lightweight model well suited for high processing rates, although it has difficulties with complex datasets. Kou et al. [7] follow this trend of detecting as fast as possible by introducing some dense layers [13] in a YOLO-v3 [14] anchor free model to deal better with scale variance in steel defects. Similarly, Zheng et al. [5] use a pre-trained SSD as a baseline to detect the anomalies in the steel sheets, looking also to minimize the processing time as much as possible. Shifting the focus to accuracy rather than speed, Liu et al. [4] use Inception Dual Network (IDN) as a detection pipeline, displaying good accuracy and processing rate. In terms of segmentation techniques, Amin et al. [8] propose a Deep Residual U-net, focusing in the global dice coefficient, as well as Boikov et al. [15], whose novel augmentation increases this specific parameter. Meanwhile, [16] use pre-trained ResNet [17] and DenseNet architectures, achieving a faster and better result than training the net with a random initialization. Bozic et al. [18] show an impressive average precision over three different datasets with an end-to-end network, performing detection and segmentation of defects. However, although all these methods are oriented to segment and classify the defects with pretty good results, still lacks a more detailed information of a per-class classification and/or segmentation.

B. Image Augmentation

When dealing with highly imbalanced datasets, the strategy to follow often relies in classical augmentation techniques such as cropping, rotation or color jitter. GANs are a good alternative to these methods, entailing more complexity to obtain pretty realistic augmentations. Either for medical [19], [20], structural inspection [21] or security inspection purposes [22], adversarial networks have shown an impressive ability to replicate patterns from real data into fake images. In the case of defect detection, Arikan et al. [23], proposed CycleGAN and pix2pix to generate non-defective and defective images respectively, by feeding the models with fake masks to generate a close-to-real defective image. However, the augmented images were contextually simple, as they feature very distinguishable background and defects, as well as a small size. Other method much more simpler but effective as well is proposed by Li et al. [24], where elongated random parts of images are cut and pasted in non-defective samples, forming anomalous data similar to scratches or dents. Nevertheless, this technique's handicap relies in the fact that could only be limited to detection of anomalies in images, unable to separate them in classes. In the case of Boikov et al. [15], a 3D modelling software is used to generate more images of different classes, rising up its dice coefficient up to 63%. It is nevertheless, as rather complex method to implement in case of being unfamiliar with this kind of tools.

III. Proposed architecture and learning strategy

In this section will be introduced the different pillars of the solution build for this work, starting with the main architecture based on a Context Aggregation Network (CAN), and the different solutions proposed to compensate imbalanced datasets.

A. Net architecture

Inspection of non-homogeneous surfaces defects may present some complications, as defects may vary substantially even inside the same class in terms of size, color or shape. That is why it is crucial to keep as much information as possible from every feature map of the network to obtain a trustworthy segmentation mask. As shown in [18] and [16], either in a encoder-decoder manner or using information from different layers of the model, it is important to keep the track of every small feature. For this purpose we propose a model that has proven high potential when it comes to detect small objects in a relatively large image. Although the approach is different in the work presented by Wang et al. [25], as they are using two CANs with different dilation factors as a cGAN and a simple discriminator, the fundamentals of their generators remain intact in our model.

As shown in Figure 2, the dilation layers go from 1 to 64 and then back to 1. The advantage of using this kind of convolution layers is that the size of all the feature maps, from the shallower layer to the deepest, remains unchanged but still captures significant information in all of them. If skip connections between opposing layers are also added, just as a regular encoder-decoder structure, then the gradient vanishing problem is mitigated at its maximum. However, the drawback is that the deeper it is, the higher the processing load will be. Thence, in contrast with the model exposed in [25],

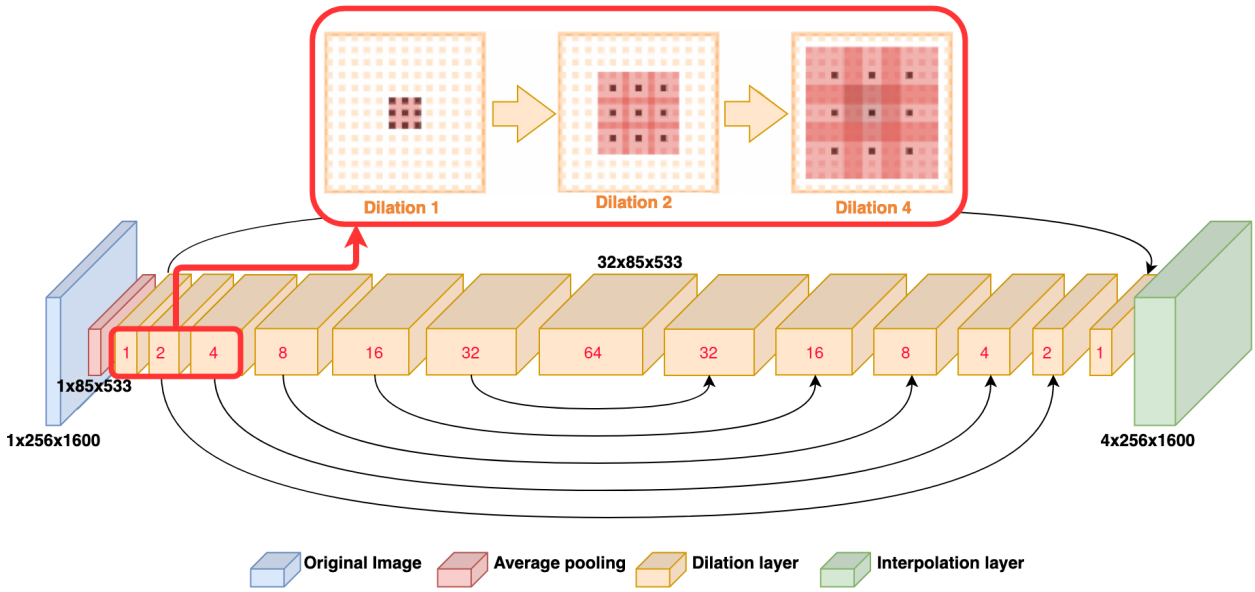

Fig. 2. CAN Architecture with the skip connections depicted with arrows and the dilation factor of every convolution layer, represented in red within each block. On the top of the image it is depicted the three first dilation layers, with a reception field of 3x3, 7x7 and 15x15 respectively.

an average pooling layer is introduced just before the first dilation. Carefully choosing the appropriate size of this filter to not vanishing the smallest defects, the new images will present the defects more gently blended with the background, favoring a more precise segmentation. Furthermore, by resizing them only at the end with a nearest interpolation, the computation time is drastically reduced compared to using full-size images. The obtained output mask is a composition of n channels, being n the number of classes to identify.

Traditionally, the classes of the defects are obtained compiling information from several layers of the model, to then concatenate and process the features with a fully connected layer [18], [16]. Regarding our case, as all the information collected from every dilation layer remains almost unaltered thanks to the skip connections, we decided to get the classes through the same segmentation masks. Taking the maximum value of every mask channel, we force the model to recognize the class, strategically linking image classification and segmentation.

B. Training policy

Having in mind the way the masks and the classes from every image are obtained, we built a loss term in such a way that the learning stage becomes progressive. It is defined as:

$$\mathcal{L}_{total} = \lambda \cdot \mathcal{L}_{class} + (1 - \lambda) \cdot \mathcal{L}_{seg} \quad (1)$$

where $\mathcal{L}_{class}$ and $\mathcal{L}_{seg}$ represent the segmentation and the classification losses respectively, with λ being a balancing parameter that depends on the current epoch of the training stage evolving this way:

$$\lambda = \frac{n}{epochs} \quad (2)$$

being n the current epoch and $epochs$ the total number of epochs. As proposed in [18], in order to make the training of the network more stable, it is convenient to give priority to the classification stage at first. That way, as the class is obtained from the same segmentation mask, the first stage will focus on identifying the defective image, while in the second part the issues are located. Without this blending of losses, the training might stuck at some point, unable to correlate both losses.

We decided to use the classical binary cross-entropy loss (BCE) for both terms of the objective function, with this common structure:

$$\mathcal{L}_{n,c} = p_c y_{n,c} \cdot \log(x_{n,c}) + (1 - y_{n,c}) \cdot \log(1 - (x_{n,c}) \quad (3)$$

where c is the corresponding class (0 or 1 as it is a binary problem), n is the number of sample of the batch and p_c the weight of the positive answer for the class c. This last term adjusts the weight for the positive samples, so the recall and the precision obtained eventually are leveraged. The bigger the value, the more biased is the loss to go towards improving the recall. As seen multiple times in this work, having an imbalanced dataset might be complicated to deal with, especially when the amount of defective pixels and classes is well below the number of non-defective ones. Due to the possible variability of sizes and quantity of the least common defects, neglecting this term would lead to disregard them in the segmentation and classification stage. Thus, it becomes necessary to carefully balance every class weight in both loss terms.

C. Guided-Crop Image Augmentation

Although the previously described losses help the model to identify better the less common and complicated classes, balancing the whole dataset is still a non-trivial task. Some classes are highly correlated to certain backgrounds and configurations or do not have too much presence in the dataset. Then, it is reasonable to think that another extra balancing tool will be needed to cope these issues. At first glance, image

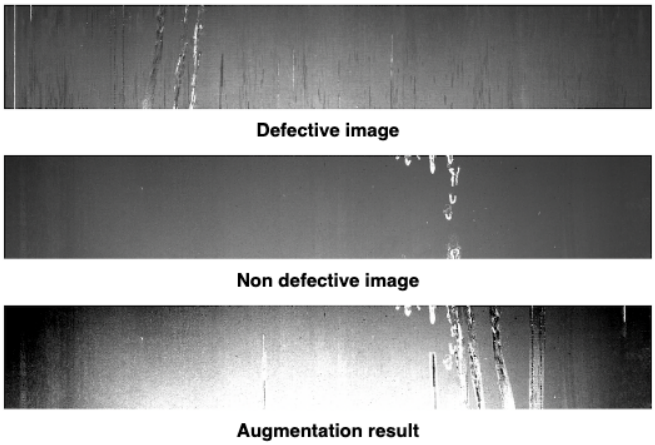

Fig. 3. Process of creation of new defective samples. First, the desired defective sample is picked (top image), then it is randomly pasted into a non defective one (middle image) and eventually tweaked with different brightness, contrast and horizontal and vertical scroll, resulting in a unique brand new image (bottom image).

augmentation could bring a little balance with techniques like rotation or color jitter, proven to be useful augmentation tools. However, this will turn insufficient as the dataset becomes more complex and imbalanced.

The solution, as proven in [23], [26], will have to be a little more sophisticated. GANs could have been considered an option if were not for the high complexity of the images, being more suited for even backgrounds and relatively small images. Besides, having very few images of some classes could also led to the overfitting of the generator. In a much simpler way and following the steps of [24], we came up with a simple but quite effective method to generate synthetic images from existing samples.

Starting from a given class, the algorithm decides how many defects (up to three) and their respective class, following the actual configuration of the dataset compiled in a lookup table. Once the classes have been decided, a random image (or images if there are more than one class to be augmented) belonging to the train set is selected, as well as a random non-defective image.

Having the defective and non-defective images already selected, a random number of the defective blobs are cropped from the anomalous ones, following the ground truth mask. In case there is only a single defective blob, then this is the only crop to perform. That way, we increase the combinations of images by the number of blobs picked from every defective sample. Afterwards, those cropped defects are tweaked (using the color jitter operation) and displaced randomly along the whole non-defective image surface to be pasted eventually in it. Previously, the crops from different classes have been assigned a priority randomly, so in case they overlap, one stays in front of the other one. This will prevent the bias of some certain classes while training the model.

Once the new image is assembled, five extra operations are performed to add even more variation with respect to the original dataset. The first two operations, vertical and horizontal random flip, are some classic techniques widely used in data augmentation. The last two operations are the vertical and horizontal displacement in the x and y axis. These last operations will only be performed if the corresponding base image has no border at the top, bottom, right or left. If this condition is ignored, then the resulting image could end up with empty spaces in the center of the image, which is never the case in the original dataset. Eventually, a color jitter is applied again to the whole new image.

The final result of the images are something similar to what is depicted in Figure 3, where a real defect is embedded in a non-defective class. Although it seems a bit obvious that it is not a real image, the several training processes that have been carried out show a step forward in terms of dice coefficient (both general and single class) and recall. Not just as the way the images are obtained, but also the way they are introduced into the train set.

This augmentation method is triggered every certain amount of epochs, where depending on the per class loss term obtained in the validation stage, probabilities will be assigned to each defective class, determining the number of instances in the new set of artificial images. For instance, if class x yields a high loss term during the validation phase compared to the rest of the classes, it will be more likely to have more representation within the new augmented dataset, having the highest probability assigned. It is important to note that it is not necessarily the minority classes that are more likely to be increased, but rather the higher-loss classes. Thus, the new training set will be composed by the original one and n new images augmented with this policy. Selecting when to perform the updates, is some kind of trade off between letting the model learn the features of the new images and not biasing excessively the predominant classes. Optimally, the model will succeed in decoupling the typical backgrounds from every class, as well as augmenting the occurrences of the most problematic classes: the smallest and the least common ones.

IV. Experimental Results

In this section, we check the performance of our approach through two challenging sets of industrial images, starting with the Magnetic Tile [11] dataset and eventually with the Severstal Steel Defect [10] dataset. We also contrast the results of our baseline against other segmentation methods, such as U-Net with Imagenet pre-trained ResNet-50 and DenseNet-121 as encoder and proven state-of-the -art models such as HRNetV1 [27] or UPerNet [28].

A. Performance metrics

Having to evaluate both classification and segmentation for each model, we considered relevant the use of the average precision and (AP) as well as the intersection over union (IoU) metrics, respectively for each task. Additionally, seeking more clarity in the results, we split this metrics between the whole dataset and the defective classes only. Eventually, we also include the processing capacity of each model in FPS, as a

TABLE I
PERFORMANCE METRICS ON MAGNETIC TILE DATASET IN TERMS AVERAGE PRECISION (AP) AND INTERSECTION OVER UNION (IOU).

Baselines	Classification		Segmentation	
	mAP (all classes)	AP (defective classes)	mIoU (all classes)	IoU (defective classes)
U-Net (Resnet-50)	94.13	[0, 0, 11.1, 0, 25.0]	**93.89**	[0, 0, 0, 0, 0]
U-Net (Densnet-121)	93.65	[11.1, 5.5, 0, 0, 0]	**93.89**	[0, 0, 0, 0, 0]
HRNetV1 [27]	95.09	[22.2, 11.1, 33.3, 0, 75.0]	93.77	[0, 0, 0.03, 0, 0]
UPerNet [28]	95.09	[22.2, 22.2, 33.3, 9.1, 100]	92.7	[0, 0, 0.6, 0, 0]
CAN (No augmentation)	96.05	[44.4, 27.8, 44.4, 81.8, 25.0]	92.34	[0, 0, 0.03, 0, 0]
CAN (500 images)	**98.32**	**[100, 94.4, 100, 100, 100]**	89.51	[30.1 39.5 57.7 35.9 60.5]
CAN (700 images)	96.17	**[100, 94.4, 100, 100, 100]**	81.33	**[32.2, 36, 57, 39.1, 58.6]**

metric to check if the model can comply with industry-like speeds.

B. Experimental setup

The implemented model and the different networks used in the different benchmarks, have been implemented in Pytorch, using two NVIDIA QUADRO P5000 in parallel. In all of them, we used the Adam optimizer with learning rate η of $5 \cdot 10^{-4}$, β_1 of 0.99 and β_2 of 0.99. For training, validation and test, we split both datasets in 75%, 12,5% and 12,5% respectively, using all the available images, performing horizontal and vertical rotation in training set. The threshold is set to 0,5 to calculate segmentation metrics. Every model that includes a dynamic augmentation policy is updated every 15 epochs.

C. The Magnetic Tile dataset

The first comparison between methods is performed using the Magnetic Tile [11] dataset, which features two of the main characteristics of an industrial application: scarcity of faults and low contrast between background and the actual defects. From the 1344 images of the image set, 392 belong to defective classes: Blowhole (115), Crack (57), Fray (32), Break (85) and Uneven (103). All the defective images come in different sizes and with their own annotations.

Using the described hyper-parameters from the section above, we trained all the models a total of 100 epochs, except for the pre-trained models that were limited to 15 as they are already trained. As for our Network, we implemented 3 different variants with loss leveraging: one without positive weighting and no augmentation, one with positive weighting and 500 dynamically augmented images and another equal one but with 700 augmented images. This last two have positive weights of [3, 3, 7, 8, 4] for classification and [18, 13, 18, 13, 7] for segmentation. That way, it is measured the impact of every contribution introduced in this work. To standardize the images, we also applied a resize of 300x400 pixels.

The key metrics of this comparison are depicted in Table I. At first sight, it is quite noticeable the difficulty that this dataset entails when approaching the segmentation problem, as seen in the IoU term for every defective class. It is not until we introduced in our model the positive weighting in both loss terms (based on every class occurrence in the dataset), when it is shown a substantial improvement in all classes, although the general mIoU lowers up a little. As for the classification terms, our CAN with 500 augmented images accomplishes

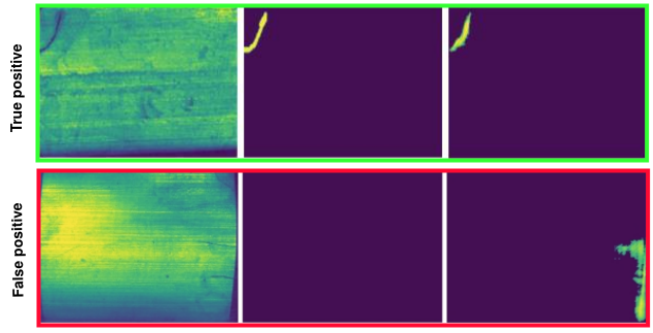

Fig. 4. Examples of Magnetic tile images with their ground truth and the output of the segmentation model.

98% mAP, as well as almost 100% AP on the faulty images. Although UPerNet [28] model sticks among the best models that do not use positive weighting, still performs worst than our baseline model with no weights. Figure 4 shows a sample of a well classified defect and a false positive from the 500 images CAN.

D. The Severstal Steel dataset

The next and last evaluation of baselines is carried out with the Severstal Steel defect [10] dataset. With similar features as the previous one, this set consist of 12568 images of 256x1600 pixels, from which 6666 include at least one defective region and up to three. the supplier distinguish four unique classes: class 1 (897), class 2 (247), class 3 (5150) and class 4 (801).

Using again the same loss and optimizer configuration, we trained every single baseline 150 epochs (due to the considerable amount of images), except again for the pre-trained models, which converge at the 10th epoch. In this case, the our approach is evaluated in 4 different baselines: one without positive weighting and no augmentation, one with positive weighting and 750 dynamically augmented images and two more with a 1000 and 1500 more images respectively. This time, the positive weighs are [4, 3, 2, 2] for classification and [8, 5, 2, 3] for segmentation.

As shown in II, still there is a major problem to segment the images properly. In fact, as class 3 is the most predominant, the segmentation metrics for this particular class are not as adversely affected as the others, with almost every model generalizing for this kind of defect. However our model, even without any kind of weighting, still achieves significant IoU terms in all classes. Even more when the dynamic augmentation is applied, reaching a 97.81% mIoU using 750 images. In the case of using 1000 and 1500 images, the general term is

TABLE II
PERFORMANCE METRICS ON SEVERSTAL STEEL DATASET IN TERMS AVERAGE PRECISION (AP) AND INTERSECTION OVER UNION (IOU).

Baselines	Classification		Segmentation	
	mAP (all classes)	AP (defective classes)	mIoU (all classes)	IoU (defective classes)
U-Net (Resnet-50)	97.63	[78.3, 76.7, 91.5, 64.4]	83.51	[0, 0, 40.3, 0]
U-Net (Densnet-121)	93.75	[97.2, 46.5, 65.5, 22.2]	80.75	[0, 0, 24.5, 0]
HRNetV1 [27]	94.24	[0, 0, 77.91, 36.67]	85.25	[0, 0, 3.1, 0]
UPerNet [28]	96.67	[40.6, 9.3, 89.4, 64.4]	87.07	[0, 0, 18.3, 2.6]
CAN (No augmentation)	98.82	[91.5, 90.7, 96.1, 90.0]	88.43	[16.3, 3.2, 38.7, 28.4]
CAN (750 images)	**99.1**	[99.1, 97.7, 98.9, 95.6]	**97.81**	[29.8, 27.9, 44.1, 38.3]
CAN (1000 images)	98.39	[95.3, 86.0, 98.9, 97.8]	86.68	[23.4, 24.8, 47.1, 41.6]
CAN (1500 images)	97.98	**[100, 97.7, 98.3, 95.6]**	84.56	**[29.6, 28.0, 44.9, 43.9]**

TABLE III
SPEED AND COMPUTATIONAL METRICS FOR EACH BASELINE.

Baselines	Speed (FPS)	Computational complexity	
	Severstal/ Magnetic	Multiply accumulate	Parameters
U-Net (Resnet-50)	32/55	14.87 GMac	11.69 M
U-Net (Densnet-121)	40/81	23.52 GMac	7.98 M
HRNetV1 [27]	14/60	102.19 GMac	1.54 M
UPerNet [28]	10/14	284.92 GMac	126.08 M
CAN	145/250	5.11 GMac	112.29 K

lowered in benefit of increasing the per-class value of IoU. When it comes to classification skills, except for the HRNetV1 [27], all the other models perform properly, being the U-Net with the Resnet-50 encoder the more balanced among the alternative baselines and a 96.63% of mAP. Nevertheless, again this model is surpassed by our standard CAN (98.82% of mAP), and one more time by the augmented ones (99.1% mAP in the 750 image augmentation). Figure 5 depicts a segmentation example from the 1000 augmented images CAN.

V. CONCLUSIONS

In this work we presented an alternative approach to image segmentation and classification, proving its effectiveness in two challenging industrial datasets [10], [11]. Using a CAN, we were able to retain as much information as possible from different dilation layers and thus helping serving its purpose in a suitable manner. We additionally introduced a dynamic augmentation policy, in order to let the model learn the features of the least common classes, tending to generalize to the whole dataset instead of just the dominant classes and null images. Together with training the model using a progressive loss leverage, to help classify at the beginning and segment at the end, we were able to overcome highly tested models like ResNet-50 and DenseNet-121 and proven state-of-the-art models like HRNet [27] and UPerNet [28] in datasets like Cityscapes [29] and ADE20K [30].

We demonstrated the strong capacity of the model to classify, accomplishing a top score of 98.32% mAP in the Magnetic tile [11] dataset and 99.1% mAP Severstal Steel defect [10] dataset. In the case of the segmentation metrics, although in [11] the Resnet-50 encoder gets the best mIoU score, our CANs with augmented images are the only ones that are able to segment properly all classes. In the case of [10], the 500 augmented images CAN tops the chart with 97.81% of mIoU. Moreover, Table III sheds more interesting

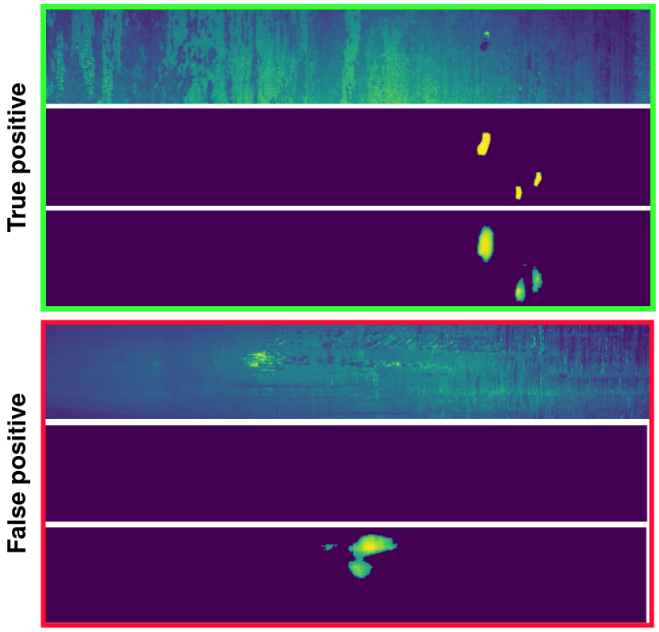

Fig. 5. Examples of Severstal steel images with their ground truth and the output of the segmentation model.

data, proving that with only 112.9K parameters, our approach overcome the other analyzed baselines, yielding frame rates from 145 to 250 FPS. Thus, our model comply with industry standards in terms of speed.

Figures 4 and 5 show a strong abbility to segment small defects, but also the trade-off that comes from augmenting artificially images, which is the rise of false positives along the images. This is why, despite having better individual IoU numbers with more augmented images, it is convenient to pay attention to this parameter in search of a better balance overall.

In view of the above, we demonstrated the applicability of our method, especially for industrial applications, when facing a hugely imbalanced dataset. The simplicity behind the augmentation algorithm, being an almost completely automatic process, makes this method potentially applicable to complex inspection tasks in the industry, both in processing speed and accuracy, with minimum changes in the configuration.

ACKNOWLEDGEMENTS

This work is partially funded by the Departament de Recerca i Universitats of the Generalitat de Catalunya under the Industrial Doctorate Grant DI 2019-40.

REFERENCES

[1] T. Nakazawa and D. V. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 2, pp. 309–314, 2018.

[2] J. Villalba-Diez, D. Schmidt, R. Gevers, J. Ordieres-Meré, M. Buchwitz, and W. Wellbrock, "Deep learning for industrial computer vision quality control in the printing industry 4.0," *Sensors (Switzerland)*, vol. 19, no. 18, pp. 1–23, 2019.

[3] G. Fu, P. Sun, W. Zhu, J. Yang, Y. Cao, M. Y. Yang, and Y. Cao, "A deep-learning-based approach for fast and robust steel surface defects classification," *Optics and Lasers in Engineering*, vol. 121, no. February, pp. 397–405, 2019. [Online]. Available: https://doi.org/10.1016/j.optlaseng.2019.05.005

[4] Z. Liu, X. Wang, and X. Chen, "Inception Dual Network for steel strip defect detection," *Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control, ICNSC 2019*, pp. 409–414, 2019.

[5] W. Zeng, Z. You, M. Huang, Z. Kong, Y. Yu, and X. Le, "Steel Sheet Defect Detection Based on Deep Learning Method," *10th International Conference on Intelligent Control and Information Processing, ICICIP 2019*, pp. 152–157, 2019.

[6] K. Liu, N. Luo, A. Li, Y. Tian, H. Sajid, and H. Chen, "A New Self-Reference Image Decomposition Algorithm for Strip Steel Surface Defect Detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 7, pp. 4732–4741, 2020.

[7] X. Kou, S. Liu, K. Cheng, and Y. Qian, "Development of a YOLO-V3-based model for detecting defects on steel strip surface," *Measurement*, vol. 182, p. 109454, sep 2021.

[8] D. Amin and S. Akhter, "Deep Learning-Based Defect Detection System in Steel Sheet Surfaces," *2020 IEEE Region 10 Symposium, TENSYMP 2020*, no. June, pp. 444–448, 2020.

[9] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[10] "Kaggle: Steel defect detection: https://www.kaggle.com/c/severstal-steel-defect-detection." [Online]. Available: https://www.kaggle.com/c/severstal-steel-defect-detection

[11] Y. Huang, C. Qiu, and K. Yuan, "Surface defect saliency of magnetic tile," *Visual Computer*, vol. 36, no. 1, pp. 85–96, 2020. [Online]. Available: https://doi.org/10.1007/s00371-018-1588-5

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," pp. 1–13, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, 2017.

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[15] A. Boikov, V. Payor, R. Savelev, and A. Kolesnikov, "Synthetic data generation for steel defect detection and classification using deep learning," *Symmetry*, vol. 13, no. 7, pp. 1–10, 2021.

[16] P. Damacharla, A. R. M. V., J. Ringenberg, and A. Y. Javaid, "TLU-Net: A Deep Learning Approach for Automatic Steel Surface Defect Detection," pp. 9–14, 2021. [Online]. Available: http://arxiv.org/abs/2101.06915

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016.

[18] J. Božič, D. Tabernik, and D. Skočaj, "End-to-end training of a two-stage neural network for defect detection," *Proceedings - International Conference on Pattern Recognition*, pp. 5619–5626, 2020.

[19] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[20] C. Han, L. Rundo, R. Araki, Y. Nagano, Y. Furukawa, G. Mauri, H. Nakayama, and H. Hayashi, "Combining noise-to-image and image-to-image GANs: Brain MR image augmentation for tumor detection," *IEEE Access*, vol. 7, pp. 156 966–156 977, 2019.

[21] Y. Gao, B. Kong, and K. M. Mosalam, "Deep leaf-bootstrapping generative adversarial network for structural image data augmentation," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 9, pp. 755–773, 2019.

[22] Y. Zhu, Y. Zhang, H. Zhang, J. Yang, and Z. Zhao, "Data Augmentation of X-Ray Images in Baggage Inspection Based on Generative Adversarial Networks," *IEEE Access*, vol. 8, pp. 86 536–86 544, 2020.

[23] S. Arikan, K. Varanasi, and D. Stricker, "Surface Defect Classification in Real-Time Using Convolutional Neural Networks," no. 1, 2019. [Online]. Available: http://arxiv.org/abs/1904.04671

[24] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "CutPaste: Self-Supervised Learning for Anomaly Detection and Localization," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9664–9674, 2021. [Online]. Available: http://arxiv.org/abs/2104.04015

[25] H. Wang, L. Zhou, and L. Wang, "Miss detection vs. false alarm: Adversarial learning for small object segmentation in infrared images," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, no. Iccv, pp. 8508–8517, 2019.

[26] X. Feng, X. Gao, and L. Luo, "X-sdd: A new benchmark for hot rolled steel strip surface defects detection," *Symmetry*, vol. 13, no. 4, 2021.

[27] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep High-Resolution Representation Learning for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021.

[28] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified Perceptual Parsing for Scene Understanding," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11209 LNCS, pp. 432–448, 2018.

[29] Y. Yuan, X. Chen, and J. Wang, "Object-Contextual Representations for Semantic Segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12351 LNCS, pp. 173–190, 2020.

[30] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin Transformer V2: Scaling Up Capacity and Resolution," 2021. [Online]. Available: http://arxiv.org/abs/2111.09883