Training of Artificial Intelligence for Automated Data Analysis of Phased Array Ultrasound Data

Richard Rhéaume, Charles Catta
Ondia Inc., 979 avenue de Bourgogne, suite 410, Québec, Qc, G1W 2L4, Canada

Abstract:

Training an Artificial Intelligence for automated data analysis of phased-array ultrasound data is a complex task. Firstly, the source of the phased-array ultrasound data is varied instrument-wise and coming from many different types of inspection: carbon steel welds, corrosion mapping, plastic welds. Secondly, the operator collecting the data and the conditions in which the data is collected influences the quality of the data. Thirdly, the knowledge and habits of the analysts analysing the data to be fed to the AI for training has a strong influence on the way the AI will recognize patterns. All these aspects of the training must be well understood in order to train an unbiased Artificial Intelligence. In this paper, we will present the methodology and results in training Ondia's own artificial intelligence dedicated to the automated analysis of phased-array ultrasound data.

1. Introduction

Training an Artificial Intelligence (AI) to analyse phased-array ultrasound (PAUT) data from weld inspection in 3D is a complex task. The goal of our study is to better understand how adding a specific type of data, in this case austenitic steel weld datasets, to a pool of carbon steel weld datasets will influence the AI's capacity to find indications in unseen carbon steel weld datasets.

Our hypothesis is that adding 6% of austenitic steel weld indications to a pure carbon steel weld dataset of indications will improve the AI's ability to detect indications in austenitic steel scans as well as improve the AI's ability to detect indications in carbon steel scans. It is also our hypothesis that since austenitic steel weld scans have a lot more noise than carbon steel weld ones, it should force our model's parameters to be more robust to noise during training making it generally better at finding indications during inference.

2. Measuring the performance of the trained Artificial Intelligence

2.1. Dataset splitting

The data used to construct a deep learning model usually comes from two distinct datasets: the training dataset and the testing dataset. The testing dataset is used to validate the model's performance on examples it has never seen before, this process is of utmost importance to properly understand the model's behavior on the field. The training dataset is used to fit the model's parameters during training by an iterative optimization algorithm such as stochastic gradient descent, it is commonly the greater of the two splits. In the case of the current model, the training dataset is made up of 85% of the available data while the testing dataset is made up of the remaining 15%.

If a model outputs undesirable results on the testing dataset it is predominantly due to one of two phenomena: underfitting or overfitting. Underfitting occurs when the model has been undertrained or does not have enough training examples thereby hindering on the model's ability to learn from useful features in the data. Overfitting occurs when the model has been over trained causing the parameters of the model to simply memorize the training data giving the false impression of a good training to the optimization algorithm.

In order to compute accurate performance metrics, the testing dataset needs to be a representative sample of the whole data. To sample representatively we employ a stratification mechanism which ensure that the training and testing datasets have the same proportions of defect types.

2.2. Performance metrics

To assess accurately the performance of a model which outputs 3d bounding boxes, we look at three different metrics: precision, overlap and confidence levels.

Precision gives a representative value of the model's ability to find boxes on examples where it should. To compute its value, we simply take the ratio between bounding boxes the model outputted in the testing dataset and the number of bounding boxes there actual was in the ground truth. For example, if the testing set contains 150 examples of defect bounding boxes and the model outputs 120 bounding boxes on those 150 examples, then our precision would be 120/150 = 80%.

The overlap metric determines how accurate the predicted boxes are in terms of location and size. To measure a box's overlap, we compute the Jaccard index. The Jaccard index varies between 0 (no overlap) and 1.0 (perfect overlap).

Source: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

The last metric, the confidence level, is a value the model outputs for each bounding box. It is an important performance metric take into consideration since a model which outputs a wrong bounding box with a high confidence level should be considered worse than a model which outputs a lower one.

2.3. Mean Average Precision (mAP)

There exists a metric called average precision which merges precision, overlap and the confidence level into a single metric. Computing the mean of all the average precisions of each box on our testing dataset gives us back the mean average precision (mAP) which is a common metric used to compare bounding box detection models in the field of deep learning. Its value mathematically is bound between 0 and 1.0 but due to the nature of bounding boxes and the methods used to compute the metric, getting close to 1.0 becomes infinitely harder.

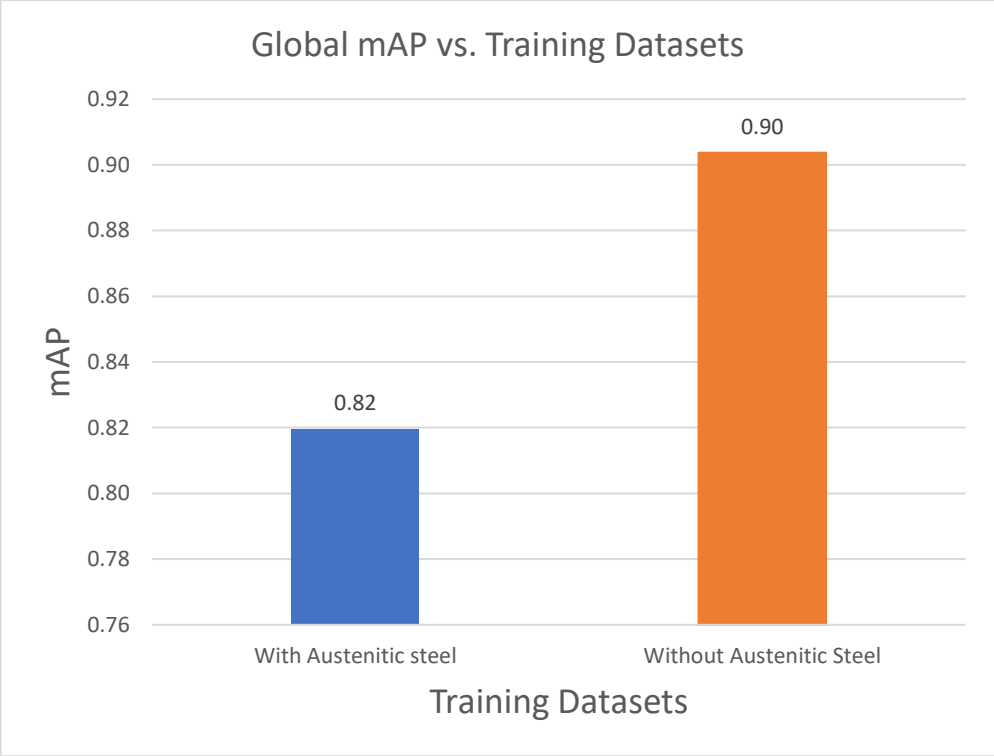
2.4. Final Training Evaluation

To compare one training to another we simply look at the shift in the mean average precision from one training run to another. It is also important to look at the shift in mean average precision by defect type as doing so might reveal potential flaws in the model. If, for example, we are comparing two models — Model 1 and Model 2 — which are both able to detect two different types of defects — Type 1 and Type 2 — and both models advertise a mean average precision of 0.6. If Model 1 detects the defect of Type 1 with a mAP of 0.9 and Type 2 with a mAP of 0.3 and Model 2 does so with a mAP of 0.7 and 0.5

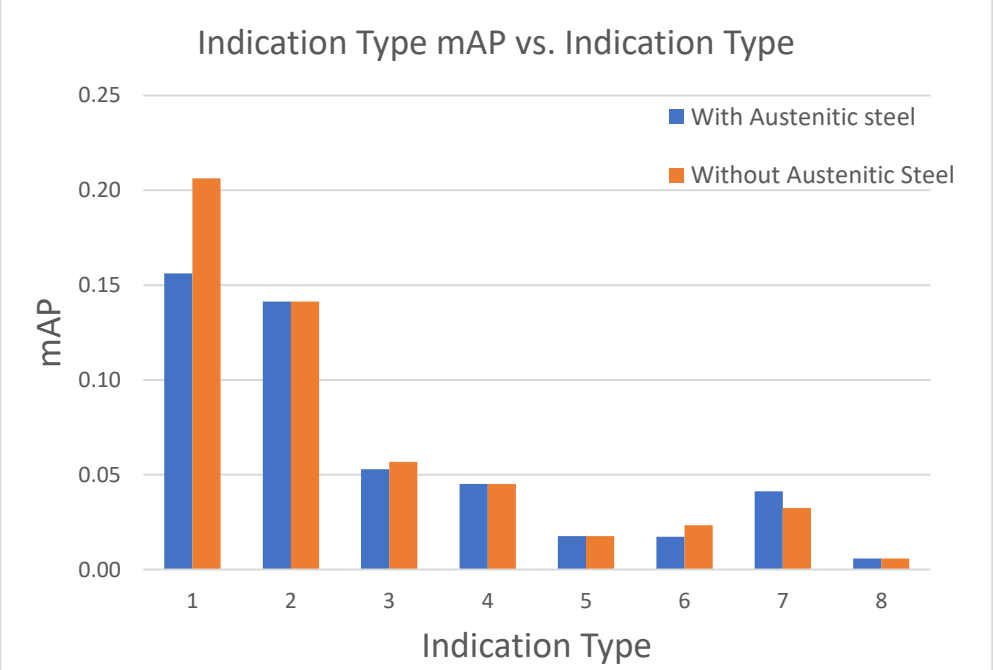
respectively, then Model 2 is a much more favorable model as it has better generalized the concept of a defect.

3. Conclusions

Contrary to our hypothesis, adding 6% austenitic steel weld indications didn't improve the AI's capacity to find indications. The AI's capacity to find indications in carbon steel weld scans is lower when austenitic steel weld indications are added to the carbon steel weld training pool as shown in the graphic 1 and 2.



Graphic 1.



Graphic 2.

This could partly be due to the nature of our austenitic data. Compare to carbon steel weld data collection, the parameterization of the instrument and the frequency of the probe are different for austenitic steel weld data collection. Frequency filters, signal smoothing and lower probe frequencies are used during austenitic steel weld data collection modifying the aspect of the indication as seen in Figures 1 – 4.

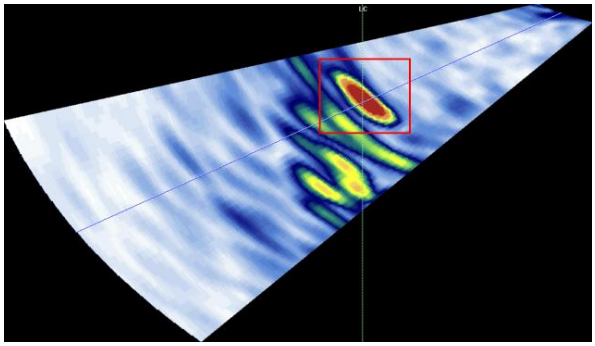


Figure 1. 2D pattern of a lack of sidewall fusion in austenitic steel

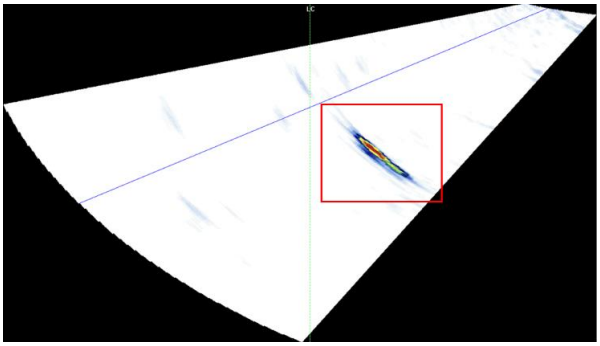


Figure 2. 2D pattern of a lack of sidewall fusion in carbon steel

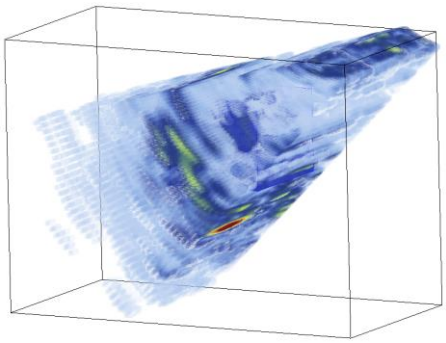


Figure 3. 3D pattern of a lack of sidewall fusion in austenitic steel

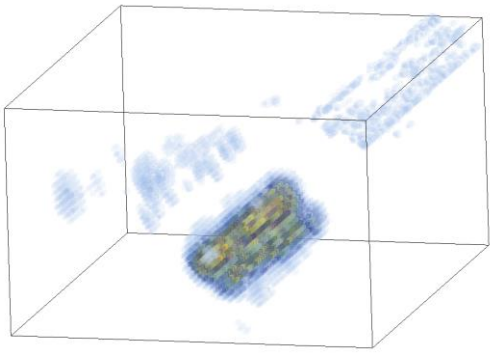


Figure 4. 3D pattern of a lack of sidewall fusion in carbon steel

Due to the fact that 3D patterns from similar indications in carbon steel welds and austenitic steel welds have major differences, it is therefore our conclusion that in order to merge austenitic steel weld indications and carbon steel weld indications in a single training pool, a larger sample of carbon steel weld indications as well as a larger proportion of austenitic steel weld indications is needed.