Real-time Cross-dataset Quality Production Assessment in Industrial Laser Cutting Machines

Nicola Peghini
Department of Industrial Engineering
University of Trento
Trento, Italy
Email: nicola.peghini@unitn.it

Andrea Zignoli
Department of Industrial Engineering
University of Trento
Trento, Italy

Davide Gandolfi
Adige Spa
Levico Terme
Trento, Italy

Paolo Rota
Department of Information
Engineering and Computer Science
University of Trento
Trento, Italy

Paolo Bosetti
Department of Industrial Engineering
University of Trento
Trento, Italy

Abstract—**In laser cutting processes, cutting failure is one of the most common causes of faulty productions. Monitoring cutting failure events is extremely complex, as failures might be initiated by several factors, the most prominent probably being the high production speeds required by modern standards. The present work aims at creating and deploying a classifier able to assess the status of a production cutting quality in a real-time fashion. To this aim, multiple datasets were collected in different environmental conditions and with different sensors. Model inputs include photo-sensors and production parameters. At first, different algorithms were tested and rated by prediction ability. Second, the selected algorithm was deployed on a GPU embedded system and added to the current machine configuration. The final system can receive the input data from the sensors, perform the inference, and send back the results to the computer numerical control. The data management is based on a client-server architecture. The selected algorithm and hardware showed good performances despite multiple changes in the environmental conditions (domain adaptation ability) both in terms of prediction ability (accuracy) and computational times.**

I. Introduction

Production monitoring is the practice of producing and using actionable information on the status of a manufacturing process to improve production quality. Highly efficient coupling between data acquisition (sensors) and information management (data processing) can prevent production system failures, low quality, or inefficient production lines.
In recent years, in several industrial applications, data availability has been surpassing the capability of the data processing systems, therefore calling for a more efficient/informative data usage and more powerful processing units. The development of new deep learning algorithms for a wide variety of industrial applications, and the concomitant spread of newer, faster, and cheaper GPU units, expanded the frontiers of the production monitoring capabilities.
Laser cutting uses a high-power laser to slice materials (e.g. metal parts, sheets, tubes, etc.) and constitutes a relatively new technology in the vast world of industrial manufacturing. Laser cutting machines, by their nature, are equipped with a high degree of automation, as both the laser optics and the computer numerical control (CNC) that move all the machine axes are typically automatically controlled. During cutting, the laser beam melts and vaporizes the material, which is eventually carried out from the cutting kerf by a jet of gas (e.g. N_2).
Several parameters/variables (such as material, laser power, gas type/pressure, cutting speed, etc.) are used to control the cut through the processed metal, and to obtain an edge with a high-quality surface finishing. However, some uncontrollable environmental and material variables can unexpectedly affect the quality of the process. These include local material property changes (e.g. due to different material batches, preprocessing, welding, polishing, deposition, rusting, etc.), external temperature/humidity, defects in the optical fiber (i.e. dirt), and geometrical inaccuracies (inconsistencies or misalignment between the computer numerical control and the actual material shape). As a consequence, the laser cutting action might not always provide a satisfactory result, leaving portions of uncut material or undesired burr made of material correctly melted but not effectively evacuated. Bad processing results are often spotted only after the end of the whole process, leading to a lot of scrap material and time-wasting. This problem is even more exacerbated in case of unsupervised and fully automated cutting operations, e.g. during night shifts.
Being able to automatically and timely detect failures during the laser cutting operations can save a considerable amount of time, energy, and material during the production process. The control loop implemented in the CNC usually takes limited information into account, e.g. distance from the material surface from proximity sensors and axis positions from encoders. Photodiodes can be used to assess the status of the cutting operations, by measuring the intensity of the light reflected by the material surface at different wavelengths.

The correct and reliable interpretation of the photodiode signals would help to complement and fusing the feedback information to be provided to the CNC. If this feedback information is timely enough and it is promptly available, then the CNC can be correctly controlled before the production process is compromised.

In a recent research work, Santolini et al. [1] shown that machine learning techniques (such as Gaussian Mixture and Hidden Markov Models) and deep learning techniques (such as Recurrent and Convolutional Neural Networks) can be used to classify the quality of the laser cutting process by fusing the sensors' information collected from the photodiodes, the laser beam source, and the CNC. However, the work of Santolini et al. [1] suffers from two major limitations: 1) an accurate classification can be only obtained on a single material/thickness couple at a time (therefore requiring new supervised training for new materials and cutting conditions), and 2) the time needed to retrieve the result of the inference is not suitable for real-time applications and industrial processes.

Therefore, the goal of this new research work was two-fold: 1) to design a new classifier able to accurately detect laser cutting process failures in multiple materials without requiring custom-built labeled datasets, and 2) to deploy the classifier and perform the inference within an extremely limited amount of time.

II. Problem Description

Similarly to [1] the model must be able to make a distinction between three classes:

1) CUT: overall good cut production quality;
2) PLASMA: a condition where cutting still occurs but the quality is lower than the standard;
3) WELDING: a condition where the final part will still be attached to the scrap material either because the laser beam is not cutting through the material or the gas flow is not enough to evacuate all the molten material.

In laser cutting production lines welding conditions are just not acceptable, while plasma conditions -still suboptimal- might only be accepted in a narrow band of circumstances. These conditions can be assessed by visually inspecting the cut surface quality.

The system has to elaborate data coming from the different sensors of the laser cutting machine and predict the cut quality (i.e. the corresponding class) that occurred in the analyzed time frame. The sensors collect data every $25\,\mu$s, and a buffer characterizing a time frame of $20\,\mathrm{ms}$ is filled. Once the buffer is ready, the classifier predicts the cut quality. The data buffer is updated every $5\,\mathrm{ms}$, after the first inference, by eliminating the oldest 200 sensors' measures and filling the buffer with the data of the most recent $5\,\mathrm{ms}$. These time frames were set by previous analysis [2]. To reduce the latency of the system to the minimum, the communication of the data buffer and the inference process has to occur under $5\,\mathrm{ms}$, so the prediction is available before the next buffer is sent to the inference server. This method will leave the system with a latency of $5\,\mathrm{ms}$ derived from the data acquisition process required to fill the buffer.

Since the classifier must perform under the $5\,\mathrm{ms}$ target time, the model needs to have an adequate number of parameters. In addition, the minimal accuracy should be set at the level of the state-of-the-art solutions, i.e. $\sim$82% [1]. Model design and development therefore must consider a trade-off between model complexity and model computational performance. To further improve performance, since some sensors' data are redundant with one another, some signals are merged to preserve all the information but lower the input dimensions.

One of the big challenges of this project was that the two datasets were collected with two different sets of photodiodes, this resulted in different raw signals which represent the same conditions. The rest of the paper is structured as follows: Section III plots an overview of the scarce literature related to the topic of this work. Section IV shows an extension to [1] comparing a few different architectures (i.e. Section IV-A), it also describes the adopted training procedure to align datasets acquired in different environmental conditions (i.e. Section IV-B). Section V describes the protocols and the hardware chosen for the deployment. In Section VI we describe the dataset used for the experiments and we also present qualitative and quantitative results that are discussed in Section VII. Finally conclusions are drawn in Section VIII.

III. related Works

Machine learning in industrial applications. Modern industrial technologies are starting to incorporate machine learning in nearly every aspect of the production process: from product inspection to quality control, from failure prediction to digitization of the industrial documentation. Importantly, machine learning enables predictive monitoring, with machine learning algorithms forecasting system failures, anomalies, breakdowns and poor production outcomes [3], [4]. In particular, in laser cutting processes, machine learning algorithms (e.g. support vector machine and artificial neural networks) have been applied to classify the quality of the production process and to help defining the optimal production parameters [5], [6]. Machine learning has also been adopted to predict bad cutting surface quality in the so-called *heat affected zones* [7] and in the dross attachment [8]. However, to the best of our knowledge, the reliability of the aforementioned models have been rarely assessed on the field. In addition, the models were not directly deployed on the industrial machines and they were not always tested for computational times. Clearly, the real-time assessment of the production quality in laser cutting production lines is still challenging [9], [10].

Domain Adaptation. Despite the blowing scientific interest in the topic [11]–[14], there are not many industrial projections of such amount of knowledge. A possible explanation is, as claimed in [15], industrial applications may have very different data distributions according to a multitude of different scenarios, depending on the intrinsic difference between sensors and the uncertain definition of a specific task. This is an observation that we share with the authors of [15] and we will briefly discuss it in Section VII. Differently to [15], however, we face a typical domain adaptation problem where the classes of the source dataset are the same as those in the target dataset, moreover we do not use adversarial learning but we try to align the distributions by considering separate statistics for different datasets.

Autonomous Driving is one of the most promising industrial application where domain adaptation is applied [16]–[18]. However, such task disposes of a multitude of large scale video datasets, in our case we only have a limited amount of self-produced data. Other works are worth to be cited but still related to generic problems such as NLP (e.g. recommendation systems [19]) or medical imaging [20].

Deployment of Deep Learning Models. Of great interest for the industrial applications, are the inference computational times of different models on different hardware solutions. For example, in [21], computational time and memory usage are reported for different model architectures and for different hardware solutions (e.g. NVIDIA Jetson TX1 and Titan X Pascal). When often times it is difficult to compare models across different disciplines (e.g. image recognition *vs* time-series classification), computational time and memory usage is known to be highly determined by model complexity.

IV. Inference Model

A. Architectures

In the first part of the work, a pool of network architectures has been tested. Starting from well known models adopted in machine vision application a Residual Networks [22], an InceptionNet [23] and a DenseNet [24] network were investigated.

• **ResNet**: In brief, Residual Networks (ResNet) are a specific type of architecture that is configured in blocks (i.e. residual blocks) each composed by traditional convolutional, regularization and skip layers (that enables the network to learn residual information from each block). In traditional neural networks, the output of one layer represents the input of the next layer, and all the layers are trained sequentially. The residual block allows skipping a few layers using a skip-connection created by defining the output of the block as the sum of the "traditional" layer output with the identity of the block inputs. Therefore, if the block layers do not contribute to the training process, they are skipped and the next layer outside the ResBlock is fed with the block input itself.

• **InceptionNet**: The basic layer of the InceptionNet is based on filter concatenation of three different convolution layers and one max pooling layer. These layers instead of going deeper they operate at the same level and then the concatenation of these layers is the input of the following inception block. To reduce the computational time by minimising the input dimensions, an extra 1x1 convolution layer is added before the convolution layer. This network is quite deep and thus the problem of vanishing gradient can arise, for this reason, Inception Networks have multiple heads in different parts of the network where the tasks can be regressed. The loss will thus be a weighted sum of the auxiliary losses (calculated on the auxiliary classifiers) and the final loss (in traditional networks for classification the loss is computed only at the last layer).

• **DenseNet**: DenseNet connects each layer with the following layers in a feed-forward fashion. The fundamental layers of this network are the DenseBlock and the TransitionBlock. In the DenseBlock a convolution block is performed, each time that this block is called in the DenseBlock is receiving as an input the feature maps of the previous iterations and its own feature-maps will be used as input into all subsequent layers. The transition block is used to concatenate seamlessly the DenseBlock.

In Fig. 1, the structures of the aforementioned architectures are presented. However, in the present research work, the input is not bi-dimensional as expected in the traditional computer vision application (i.e. images), but it is a set of machine signals and sensors at a specific time, so the network has to be adapted to accept time-series input shapes.

B. Adaptation

Maintaining a neural network for industrial applications that are inherently changing, updating and evolving technologies is quite challenging. For these particular applications, the bottle-neck consists of the large quantity of labelled data needed to re-train and tune the weights when the system is used under different conditions (i.e. different machine models, model updates, new materials, cutting parameters updates, environmental changes, etc.) with respect to the one used to collect the training dataset. This impediment arises the need to establish a starting point (model with labelled samples) from which it is possible to tune the weights using unlabelled data collected, for example, while setting up the machine. To this, inspired by the works of Carlucci et al. [11] and Roy et al. [12] a simultaneous training of a classification network using a labelled (source-dataset) and an unlabelled (target-dataset) was performed (Fig.2).

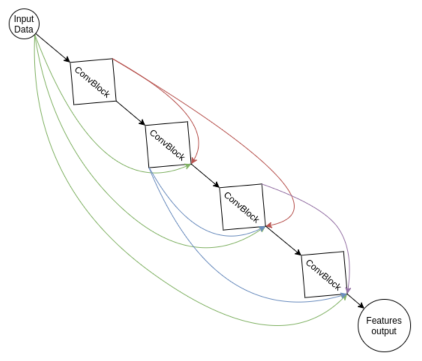
In this work we consider a labelled source dataset $\mathcal{S} = \{(x_1^s, y_1^s), \ldots, (x_n^s, y_n^s)\}$ and a non-labelled target dataset $\mathcal{T} = \{x_1^t, \ldots, x_m^t\}$. The training is performed fusing two different losses, a sparse crossentropy for $\mathcal{S}$ (i.e. Eq. (1)) and entropy for $\mathcal{T}$ (i.e. Eq. (2)).

$$\mathcal{L}_s(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \log f_s^{\theta}\left(y_i^s; x_i^s\right) \tag{1}$$

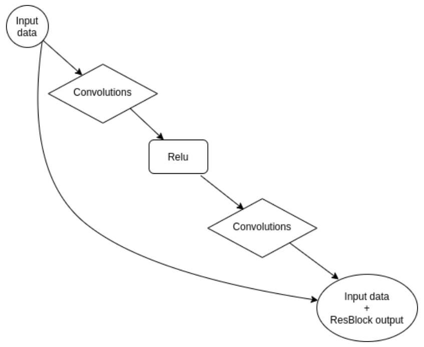
$$\mathcal{L}_u(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{y \in \mathcal{Y}} f_t^{\theta}\left(y; x_i^t\right) \log f_t^{\theta}\left(y; x_i^t\right) \tag{2}$$

The two losses have then been fused in an unique loss using a scaling factor λ as shown in (3).

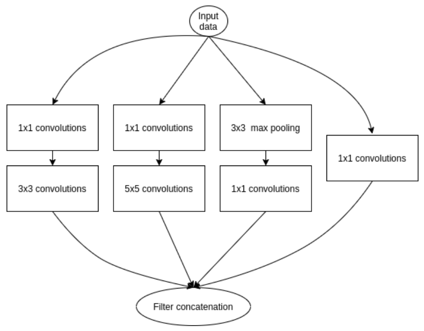
$$\mathcal{L}(\theta) = \mathcal{L}_s(\theta) + \lambda \mathcal{L}_u(\theta) \tag{3}$$



(a) Schematic representation of the DenseNet neural network characteristic layer.



(b) Schematic representation of the ResNet neural network characteristic layer.



(c) Schematic representation of the InceptionNet neural network characteristic layer.

Fig. 1: Different neural network characteristic layers.

During training, the batch normalization layer statistics were computed in a distinct manner between the two datasets, these separate statistics have an high impact on the final model result.

V. Deployment

A. Client server communication

A client/server system is used to establish the communication between the machine side (*client*) and inference hardware (*server*). The *client* is responsible for reading, storing, and buffering the sensors' information (this is made possible by a dynamic library). The *server* receives the data in the buffer and invokes a dynamic library that takes care of the inference. Once the results of the inference have been computed, they are sent back to the *client*, which uses the information thereby contained to instruct the control loop. To make the communication robust, fast, and efficient, the high-performance asynchronous messaging queuing library ZeroMQ has been used.
The sensors' information is transferred from the *client* side in binary form while undergoing a serialisation and deserialisation process designed to minimise the cycle-time. This process is supervised by an error-detecting code (cyclic redundancy check CRC8). Therefore, the buffer sent from the *client* is composed by a header of 4 bytes indicating the buffer length, this allows to check if all the information needed from the sensors are gained correctly, then all the data themselves (sensors information) and finally a check byte (CRC8) to check if any communication error occurred. The inference result availability is checked by the *client* with a polling action; if available, the predicted class is read by the *client* and thus the control system can react accordingly. On the *server* side the results are processed by the dynamic library that takes care of the inference. The dynamic library was created to leave the *client/server* agnostic to the data buffer and the deployment hardware and workflow, this approach allows the deployment on multiple platforms.
In industrial applications is always good practice to be able to store logs of the process to intervene in case any issue arises. In this case, to avoid interfering with the process and create any undesirable lag, a publisher-subscriber setup was created. The *server*, if this option is activated, will publish the inference results and any error occurring during run-time on a dedicated thread, the industrial PC will then store the log elsewhere, this setup was created to avoid clogging up the inference hardware with data.

B. Hardware

The selected hardware needed to be suitable for the real-time task required, i.e.: cost-effective but still able to retrieve and send back the result of the inference within $5\,\mathrm{ms}$; this upper limit in communication time was set to give enough time to the control loop to react and consequently modify the machine parameters. Some of the adjustments can be made in real-time but the majority of reactions are limited by the mechanics of the machine itself. A compact solution suitable for industrial applications (ZiggyBox$^{\mathrm{TM}}$, which is based on NVIDIA$^{®}$ Jetson$^{\mathrm{TM}}$TX2 computing module) has been selected for the deployment.
The inference is computed using the TensorRT$^{\mathrm{TM}}$framework, which uses an ONNX file, open format built to represent machine learning models, to decode the model. This solution allows the system to be independent of the framework used to train the network. TensorRT$^{\mathrm{TM}}$optimises the computational times for NVIDIA$^{®}$ platforms (equipped with ARM64 processing unit architectures).
With the goal to increase the performance of the hardware, automatize as much as possible the process and avoid any unnecessary communication and inference calculation time jitter, the Ubuntu GUI was deactivated. A start-up service (hardware starts when the machine is booted) was created, which activates all the CPUs, maximize energy usage and clock rate (power consumption is not an issue for this application).

VI. Experiments

A. Datasets

Two experimental campaigns were performed to collect the data in different environmental conditions and constitute two

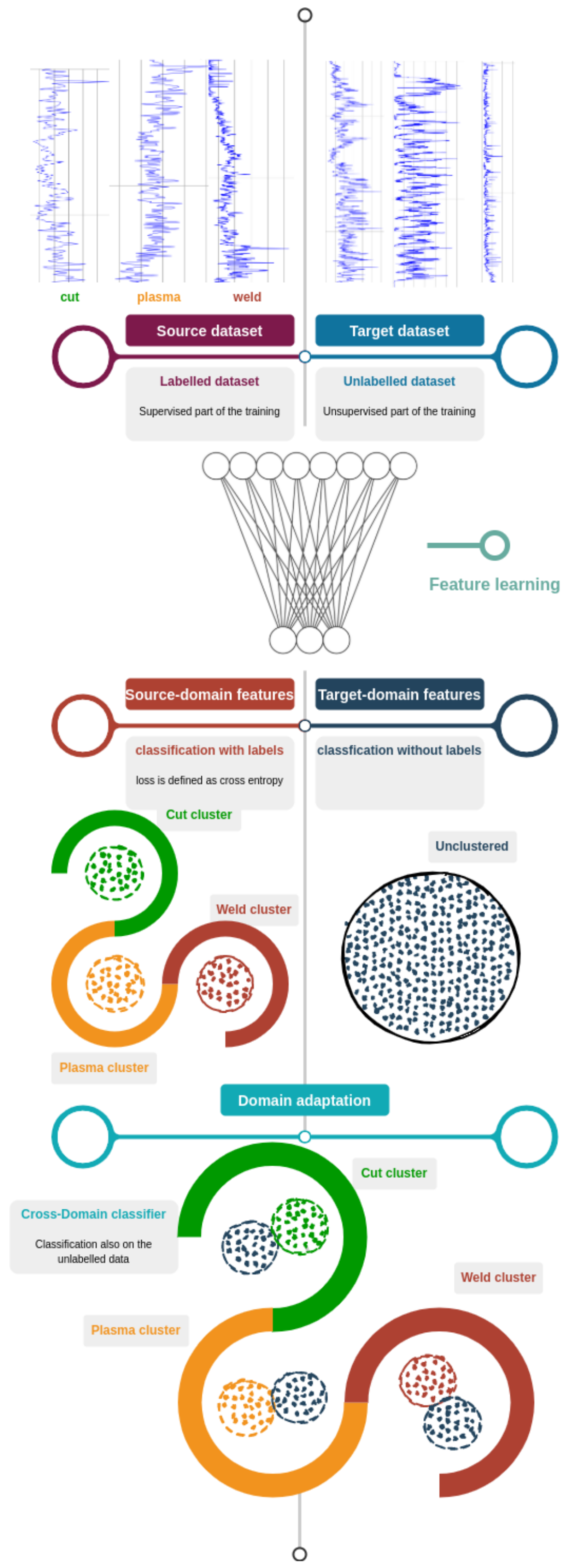


Fig. 2: Workflow for the simultaneous training of a classification network using a labelled (source-dataset) and an unlabelled (target-dataset)

different datasets (DS1 and DS2). The two data acquisition campaigns were performed on the same machine with different sensors (i.e. the set of photodiodes), different material batches and different layer thicknesses. The goal was to introduce variability in the datasets, which was used to test the reliability and the robustness of the system. The DS1 was collected in a cool environment ($\sim 18°C$) during the winter season on the following combinations of materials and thicknesses: stainless steel (1.5, 2.5, 4.0 and 8 mm), construction steel (2.3, 3.0, 5.0 and 7 mm) and aluminium (2.0, 4.0 and 8 mm). The DS2 was collected in a warm environment ($\sim 24°C$) during the summer season on the following combinations of materials and thicknesses: stainless steel (2.0, 3.0 and 6 mm), construction steel (1.8, 3.0 and 6 mm) and aluminium (1.5, 3.0 and 6 mm).

B. *Training procedure*

Eight different training/testing sessions (S1-8, table I) were performed to assess the accuracy of the different model architectures. In S1, supervised (i.e. with labelled data) training and testing were both performed on DS1 normalised on max/min values found in DS1. In S2, supervised training was performed on DS1 and testing was performed on DS2, with DS1 and DS2 both normalised on max/min values found in DS1. In S3, supervised training was performed on DS1 and testing was performed on DS2, with DS1 and DS2 normalised on max/min values found in DS1 and DS2, respectively. In S4, supervised (i.e. with labelled data) training and testing were both performed on DS2 normalised on max/min values found in DS2. In S5, supervised training and testing were both performed on an unified dataset (DS1 + DS2), which was normalised on max/min values found in the same unified dataset. In S6, supervised training and testing were both performed on an unified dataset (DS1 + DS2), where DS1 and DS2 were normalised on the max/min values found in the correspondent datasets. In S7, supervised training was performed with DS1 and unsupervised (i.e. without labelled data) training was performed on DS2. Testing was performed on the unified dataset (DS1 + DS2), where DS1 and DS2 were normalised on the max/min values found in the correspondent datasets. In S8, unsupervised training was performed with DS1 and supervised training was performed on DS2. Testing was performed on the unified dataset (DS1 + DS2), where DS1 and DS2 were normalised on the max/min values found in the correspondent datasets.

C. *Results*

1) Model accuracy: In table I the model accuracy results are presented for the ResNet, which provided the best accuracy (the DenseNet accuracy was as high as 89.48% in the case of different normalization but decreased to 87.03% in the case of domain adaptation; the InceptionNet instead showed low accuracy ($\sim$75%) in the case of different normalization of the dataset and was not tested for domain adaptation abilities).
The high accuracy obtained in S1 was not clearly maintained during S2 (88.76% *vs* 23.61%). Apparently, normalisation only played a minor but meaningful role in this loss of accuracy,

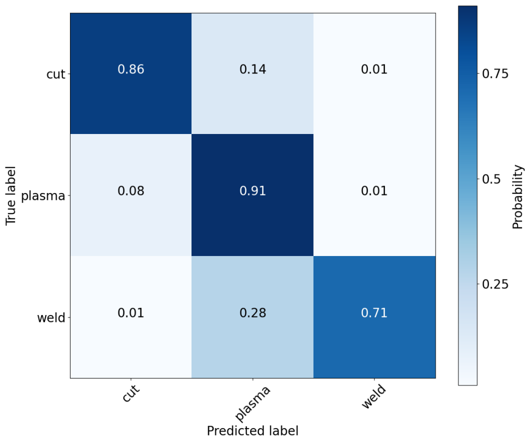
Session	Train.	Test	Norm. DS1	Norm. DS2	Acc.
S1	DS1	DS1	DS1	-	88.76%
S2	DS1	DS2	DS1	DS1	23.61%
S3	DS1	DS2	DS1	DS2	40.50%
S4	DS2	DS2	-	DS2	89.92%
S5	DS1 + DS2	DS1 + DS2	DS1 + DS2	DS1 + DS2	70.44%
S6	DS1 + DS2	DS1 + DS2	DS1	DS2	84.78%
S7	sDS1 + unsDS2	DS1 + DS2	DS1	DS2	89.92%
S8	unsDS1 + sDS2	DS1 + DS2	DS1	DS2	89.69%

TABLE I: Combination of the different dataset and normalization criteria used during training and testing, accuracy on the test dataset, the second part relate to the experiments performed using the proposed domain adaptation strategy; all the results are relative to the ResNet model.

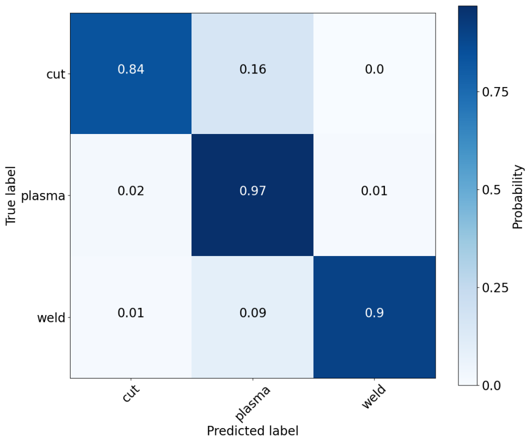
since in S3 the accuracy level was still very poor (40.50%). The accuracy provided in S4 (89.92%) confirmed that the ResNet was able to preserve a high prediction ability when the same dataset used for training was also used for testing. The accuracy levels obtained in S5 and S6 (70.44% *vs* 84.78%) revealed again the marginal but tangible contribution of the normalisation procedure. Interestingly, high levels of accuracy were maintained in the two sessions were the ResNet was tested for training adaptation abilities (S7-S8). In these two last sessions, accuracy was as high as 89.92% and 89.69%. The confusion matrices reported in Fig.3, revealed a tangible improvement when changing from the fully supervised to the partially-supervised approach.

To evaluate the network performances more practically the network predictions, labels, and sensor signals for the whole cut geometry are plotted together in Fig.4. With these plots it is possible to appreciate the accuracy for the single geometry, this allows also to analyze the physical part and identify the critical situations. The value sent back to the client after the inference is performed is an integer representing the cutting class (cut quality) not the probabilities predicted of the network. But the probabilities are also reported: these are helpful to understand if the model is classifying the cut with high certainty or if the right or wrong prediction is a consequence of the maximization of the probabilistic prediction. The model tends to exclude the more distant class, this behaviour can be appreciated also from the confusion matrices where the top-right and bottom-left corners have near-zero probability value.

2) Deployment: The communication and inference times for 12 experiments are reported in Fig. 5. Each test is performed on 10000 inferences, thus corresponding to a process elapsed time of 50 s. The average time for both communication and inference was 2.3 ± 0.15 ms, well below the target time of 5 ms. In a few occasions (in the 0.02% of the total collected samples) the inference took longer to provide the results. The communication system must also deal with these outliers: if the result is not available for a set number of consecutive cycles the connection with the server is interrupted by the client and reestablished. This solution was selected because one missed step is acceptable but if multiple predictions are missed consecutively the system will crash and so the whole process must be automatically started again.



(a) Confusion matrix of S6

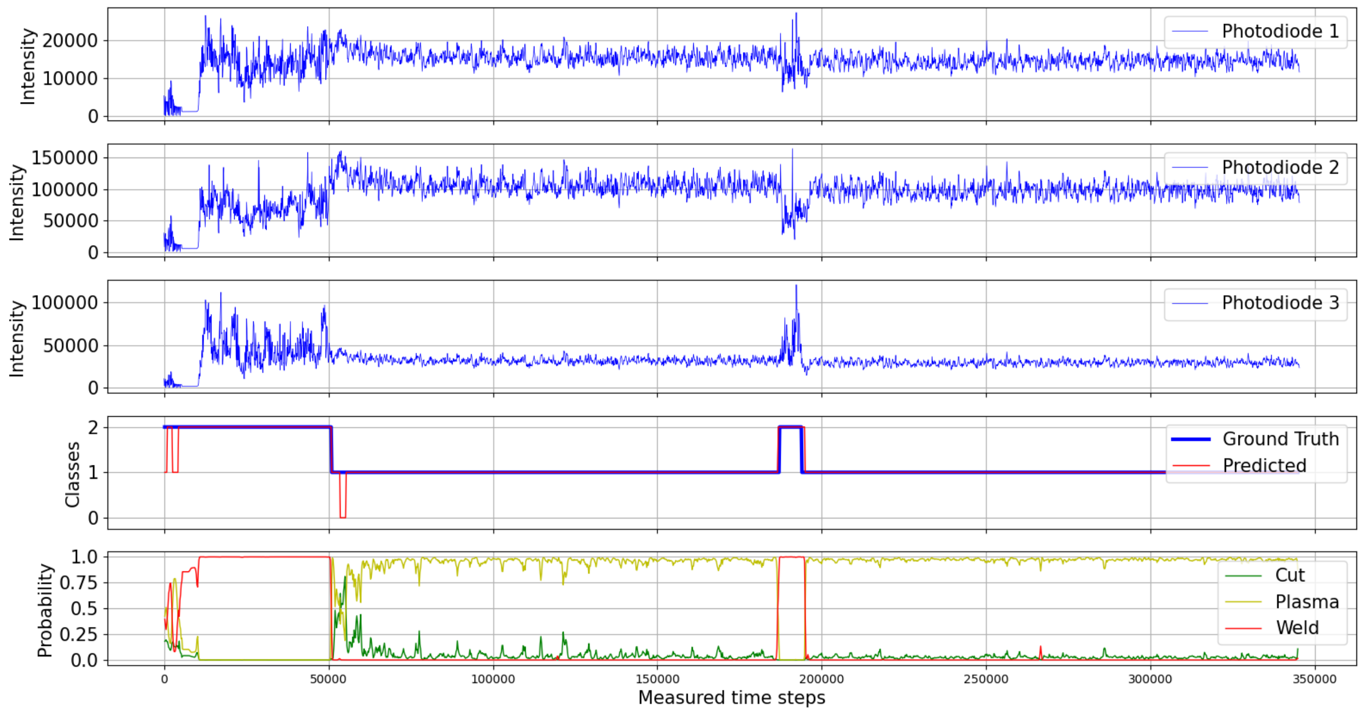


(b) Confusion matrix of S7 on testing subset derived from the target dataset

Fig. 3: Confusion matrices

VII. Discussion

Product reliability is key in industrial applications, as machines need to compensate for the different and often chaotic challenges that could occur across their entire lifespan, e.g.: software and hardware updates, overuse, wearing, environmental condition changes, etc.. The current work extends the work done by Santolini et al. [1] on quality production assessment for laser cutting machines. However, the present work is innovative in two important aspects: 1) transfer learning between labelled and unlabelled datasets and 2) real-time inference. The obtained results are very promising and show that the use of additional data, despite the lack of labelled data, is beneficial in terms of the overall accuracy of the model. The upper bound of the presented models seems to be close to 90%, and this might be due to the annotation noise implacable to an operator-dependent disagreement. Unfortunately, only a unique label set was available, so such claim remains hypothetical and speculative. An additional observation is about the normalization strategy adopted during the training/testing sessions. It was noticed that model performance improved when per-dataset



(a) Sample of 6 mm thick aluminum

Fig. 4: Inference plot, the information reported in this plots are the sensors signals (network inputs), the labels and the prediction returned to the client in order to close the information loop and finally the network predictions.

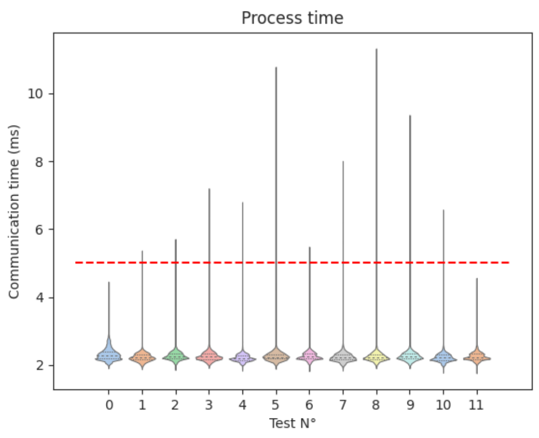


Fig. 5: The probability density of the data at different values (violin plot) is reported for a number of inference tests. Markers for the median of the data and indicating the interquartile ranges are reported. A red dashed line represents the upper limit to the computational time required (i.e. 5 ms).

normalization was applied. This is reasonable, considering the different set of sensors employed in the two datasets: in the second campaign the adopted sensor appeared much more sensitive and they resulted in a more noisy signal spanning a slightly different range of magnitudes. In addition to the adaptation procedure, this paper also shows a comparison to a set of different well-known architectures, adapted to the specific task. Densenet and ResNet resulted to have similar accuracy, however, the latter has less parameters and therefore ensures a faster inference.

Finally, an important contribution of the study is the deployment of the model on the laser cutting machine. The complexity and potentially the maximal accuracy of the neural network was restrained by the maximal time allowed for computing the inference. As previously mentioned, the time limit of 5 ms has been set *a priori*. If the neural network was able to complete the inference within this time limit, then the CNC control loop could be close in due time. In the world of practice, with CNC controlling the laser beam at speeds as high as $30\,\mathrm{m/min}$, an intervention time of 5 ms would allow sub-optimal cutting to occur for a length of about 0.5 mm (worst case scenario). Given this strict time constraint, a lot of energy has been invested in developing and optimizing a model-agnostic deployment framework, that allows deployment on multiple platforms and, thanks to the TensorRT$^{\mathrm{TM}}$framework (i.e. NVIDIA$^{\circledR}$ platforms only), to be independent of the training process. The final deployment has been made on a separated NVIDIA$^{\circledR}$ hardware, to allow easy testing and better performance on industrial computers.

VIII. Conclusion

This work proved the feasibility of a neural network real-time deployment for an industrial application. In terms of technology readiness level, this research work does not only provide a proof of concept, but a component validated under controlled laboratory conditions. The neural network deployed here can complete the inference process at the $\sim 89\%$ of accuracy within $5\,\mathrm{ms}$. The neural network has been proved to be robust against sensor and environmental changes and displayed the ability to adapt and transfer to new datasets without the need for an additional labelling campaign. This work opens up new avenues for future research in the field of partially and fully unsupervised models operating in industrial environments. However, to eliminating the labelling process, further testing is warranted.

Acknowledgment

The authors would like to thank Adige S.p.A. for the opportunity of working on such interesting research project, for their expert advice and collaborative flair. We would like to thank ProM Facility for providing computational resources and support that have contributed to these research results.
The project presented in this paper has been funded with the contribution of the Autonomous Province of Trento, Italy, through the Regional Law 6/99. Name of the granted Project: LT4.0.

References

[1] G. Santolini, P. Rota, D. Gandolfi, and P. Bosetti, "Cut quality estimation in industrial laser cutting machines: A machine learning approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[2] G. Santolini, "Deep Learning Models for Cut Interruption Detection in Laser Cutting Machines," Master's thesis, University of Trento (Department of Industrial Engineering), Trento, 2019.

[3] P. Larrañaga, D. Atienza, J. Diaz-Rozo, A. Ogbechie, C. E. Puerto-Santana, and C. Bielza, *Industrial applications of machine learning*. CRC Press, 2018.

[4] S. K. Das, S. P. Das, N. Dey, and A.-E. Hassanien, "Machine learning algorithms for industrial applications," 2020.

[5] H. Tercan, T. Al Khawli, U. Eppelt, C. Büscher, T. Meisen, and S. Jeschke, "Improving the laser cutting process design by machine learning techniques," *Production Engineering*, vol. 11, no. 2, pp. 195–203, 2017.

[6] Z. Jurkovic, G. Cukor, M. Brezocnik, and T. Brajkovic, "A comparison of machine learning methods for cutting parameters prediction in high speed turning process," *Journal of Intelligent Manufacturing*, vol. 29, no. 8, pp. 1683–1693, 2018.

[7] O. Anicic, S. Jović, H. Skrijelj, and B. Nedić, "Prediction of laser cutting heat affected zone by extreme learning machine," *Optics and Lasers in Engineering*, vol. 88, pp. 1–4, 2017.

[8] L. Franceschetti, M. Pacher, M. Tanelli, S. C. Strada, B. Previtali, and S. M. Savaresi, "Dross attachment estimation in the laser-cutting process via convolutional neural networks (cnn)," in *2020 28th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2020, pp. 850–855.

[9] U. Halm, D. Arntz-Schroeder, A. Gillner, and W. Schulz, "Towards online-prediction of quality features in laser fusion cutting using neural networks," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2020, pp. 346–359.

[10] C. Alippi, V. Bono, V. Piuri, and F. Scotti, "Toward real-time quality analysis measurement of metal laser cutting," in *2002 IEEE International Symposium on Virtual and Intelligent Measurement Systems (IEEE Cat. No. 02EX545)*. IEEE, 2002, pp. 39–44.

[11] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulo, "Autodial: Automatic domain alignment layers," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 5077–5085.

[12] S. Roy, A. Siarohin, E. Sangineto, S. R. Bulo, N. Sebe, and E. Ricci, "Unsupervised domain adaptation using feature-whitening and consensus loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9471–9480.

[13] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[14] J. Munro and D. Damen, "Multi-modal domain adaptation for fine-grained action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 122–132.

[15] X. Li and W. Zhang, "Deep learning-based partial domain adaptation method on intelligent machinery fault diagnostics," *IEEE Transactions on Industrial Electronics*, 2020.

[16] Y. Shan, W. F. Lu, and C. M. Chew, "Pixel and feature level based domain adaptation for object detection in autonomous driving," *Neurocomputing*, vol. 367, pp. 31–38, 2019.

[17] S. Zhao, B. Li, X. Yue, Y. Gu, P. Xu, R. Hu, H. Chai, and K. Keutzer, "Multi-source domain adaptation for semantic segmentation," in *Advances in Neural Information Processing Systems*, 2019, pp. 7287–7300.

[18] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2020–2030.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *ICML*, 2011.

[20] J. Dong, Y. Cong, G. Sun, B. Zhong, and X. Xu, "What can be transferred: Unsupervised domain adaptation for endoscopic lesions segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4023–4032.

[21] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.