The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-022-10516-5

ORIGINAL ARTICLE

Assessment of milling condition by image processing of the produced surfaces

Nicolas Carbone[1,2] · Luca Bernini[1,2] · Paolo Albertelli[1,2] · Michele Monno[1,2]

Received: 3 June 2022 / Accepted: 8 November 2022
© The Author(s) 2022

Abstract
The digital industrial revolution calls for smart manufacturing plants, i.e. plants that include sensors and vision systems accompanied with artificial intelligence and advanced data analytics in order to meet the required accuracy, reliability and productivity levels. In this paper, we introduce a surface analysis and classification approach based on a deep learning algorithm. The approach is intended to let machining centres recognise the adequacy of process parameters adopted for the milling operation performed, based on the phenomenological effects left on the machined surface. Indeed, the operator will be able to understand how to change process parameters to improve workpiece quality of subsequent parts by a reverse engineering procedure that reconstructs the process parameters that generated the analysed surface. A shallow convolutional neural network was proposed to work on surface image patches based on a limited training dataset of optimal and undesired cutting conditions. The architecture consists of a series of 3 stacked convolutional blocks. The performance of the proposed solution was validated through 5-fold cross-validation, measuring the mean and standard deviation of the f1-score metric. The algorithm arrived at outperformed the best state-of-the-art approach by 4.8% when considering average classification performance.

Keywords Surface classification · Milling · Neural networks · Process parameters · Inference

1 Introduction

Industry 4.0 introduced the paradigm of smart manufacturing plants and machining centres. The term *smart* underlies the capability of manufacturing technology to use advanced data analytics to improve machining performance and support the operator [1, 2]. In recent decades, many techniques were developed to deal with this goal, most of them being in the realm of artificial intelligence. Research in artificial intelligence received a boost especially due to advances in information processing technology and machine learning techniques, capable of learning models and decision rules from training sensorial or pictorial data [3]. Machine learning algorithms constituted the benchmark for the approach arrived at in this paper. With the advent of cheaper sensors, increased storage capacities and more powerful computers, deep learning started to become more attractive with the implementation of neural networks and the discovery of more effective training algorithms for them. Deep learning represented the natural evolution of machine learning, gaining the capacity to automatically identify discriminatory characteristics of signals and pictures from training data [4, 5]. Machine learning firstly, and deep learning, secondly, were involved in many fields related to image processing and classification ranging from land-use categories classification [6], object recognition and classification in road scans [7], to traffic signs recognition [8], texture analysis in turning processes [9] and surface defect recognition [10–15], as well as prediction of car prices from images [16],

✉ Luca Bernini
luca1.bernini@polimi.it

Nicolas Carbone
nicolas.carbone@polimi.it

Paolo Albertelli
paolo.albertelli@polimi.it

Michele Monno
michele.monno@polimi.it

[1] Department of Mechanical Engineering, Politecnico di Milano, via La Masa 1, 20156 Milan, Italy

[2] MUSP Macchine Utensili Sistemi di Produzione, strada della Torre della Razza, Piacenza 29122, Italy

Published online: 02 December 2022

robot assembly operations [16] and manufacturing trajectory smoothing [17].

In this paper, we propose a deep learning approach with the aim of improving the choice of machining parameters and detecting undesirable machining conditions. The innovation brought by the presented research regards the analysis of the phenomenological effects of the milling process on the machined surface. Thus, the technological footprint of the milling operation is analysed through images processing, in order to make inference on the process. The proposed approach has the goal to perform a reverse engineering task, thus it is able to reconstruct both technological and process parameters used to machine the surface. Indeed, undesired cutting conditions (i.e. tool chippings and excessive run-out) and inappropriate cutting parameters are detected through image analysis. The approach arrived at is based on a shallow convolutional neural network (CNN) capable of detecting wrong cutting speeds, feed per tooth, and machining conditions, as well as technological parameters from machined workpiece surface images. Thus, from the CNN, the operator can retrieve useful information on how to take corrective actions on process parameters to reach higher production quality levels for subsequent parts to be machined.

The structure of this paper is as follows: in Section 2, a state-of-the-art view of traditional texture descriptors, machine learning and deep learning classifiers is presented; in Section 3 the setup, the experimentation and dataset presentation are thoroughly explained, together with the definition of the f1-score performance metric; in Section 4.1, the extraction of state-of-the-art texture descriptors and application of machine learning classifiers to our problem (with benchmark purposes) are described; in Section 4.2, the CNN architecture arrived at and selection of hyperparameters are set out; in Section 5, the f1-score performance metric (mean and standard deviation) is compared when using state-of-the-art approaches and the CNN developed, together with a proper discussion of the results; finally, conclusions are drawn at the end of the paper, in Section 6.

2 State-of-the-art

2.1 Traditional texture descriptors

In order to prepare images for machine learning classification, it is necessary to manually extract features from them. Traditional texture features may belong to the spectral, structural and statistical world [9] or come from fractal analysis [18]. When dealing with milled surface images, statistical features are the most commonly used and relevant for analysis. Four of them are the most used: Grey-Level Co-Occurrence Matrix (GLCM), Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP) and Dual Cross Patterns (DCP).

GLCM is a second-order statistical texture descriptor, i.e. it accounts for the relative position of two pixels. In fact, GLCM computes the number of occurrences of pairs of pixels with a given intensity and a given displacement [9]. GLCM was introduced by Haralick et al. [6] and was widely used in many image classification and analysis problems, while being selected as a benchmark for deep learning algorithms for performance evaluation [9, 12, 13]. HOG computes the frequency of occurrence of the orientation of the gradient in a localised portion of an image. HOG was created for object detection in images and was adopted in image classification and analysis problems [5, 16, 19] (even as a benchmark). LBP computes the histogram of a transformation of 8 digit binary numbers obtained on 3x3 pixel cells of the image to decimal format. LBP was identified as one of the most used and effective statistical descriptors for classification tasks on textures. Its success was related to its invariance in relation to grayscale and rotation. Several modifications to its definition were also developed to overcome some related challenges, e.g. noise sensitivity [10]. LBP was applied in several contexts, not only related to machining: Song et al. applied it for defect recognition [10], Garcia-Ordas et al. employed it for tool wear monitoring based on computer vision [19], Fu et al. used it as a benchmark for steel surface defect classification [13] and Hou et al. used it as a benchmark for classifying cancer sub-types [20]. DCP represents a second-order image descriptor, with the goal of performing local sampling and pattern encoding. It encodes discriminatory characteristics of images in the main eight directions. DCP was born for face recognition [21], but recently found applications in the detection of steel surface defects, like in [15].

2.2 Machine learning classifiers for texture descriptors

Machine learning classifiers are based on the manual extraction of features/descriptors. Thus, GLCM, HOG, LBP and DCP are given as input to machine learning approaches to assign classes to the original data. Machine learning approaches can perform this task without being explicitly programmed, but by proper *learning* from training data [3]. Some of the most common machine learning approaches for classification purposes are k-Nearest Neighbours (KNN), Support Vector Machine (SVM) and Random Forest (RF) classifiers [22, 23].

KNN is a simple classifier based on the maximum occurring class in the k nearest training data according to a specific distance measurement [24]. It was widely used in literature for classification tasks, such as steel surface

defect detection [10, 13]. SVM finds the optimal hyperplane for separating features that belong to two or more groups. It has been applied in the medical sector through image classification [25], steel surface defects detection [10], object detection in road environments [7], tool wear classification [19] and benchmarking [13, 16, 26, 27]. RF combines a set of trivial classifiers (called decision trees) and select the output class based on a voting system [28]. RF were used in classification tasks such as object detection of RGB-D images (RGB images with depth information), unordered point set context prediction [29] and as a benchmark [5].

2.3 Convolutional neural networks are end-to-end classifiers

Deep learning methods typically assume the form of deep neural networks. Deep learning has the advantage, over machine learning, of automatically extracting discriminatory features and descriptors from training data. In this way, they work directly on raw input data/images. Nevertheless, they typically require a huge amount of training data to learn how to classify the data provided [4], thus generally preventing their use for limited datasets. The most used architecture for classification of images is the CNN [20]. CNNs are based on stacked convolutional layers that perform automatic extraction of low-level and high-level features across the layers, while simultaneously classifying images with the last one. In the recent years, there has been a succession of many *standard* architectures, trying to solve commonly faced problems and increase the performance of CNNs. The deeper the network is, the more accurate and powerful it is. Based on this evidence, He et al. [30] introduced the so-called Residual Neural Networks (ResNets). ResNets made it possible to overcome the problem related to the fact that a deeper and more sophisticated network architecture may collapse on its shallower counterpart simply with a layer learning the identity function [30]. Facing the gradient vanishing effect problem during training of deeper and deeper CNNs, Huang et al. proposed Dense Networks (DenseNets) by connecting multiple layers of CNNs via short paths [31]. This architecture reached ResNets' performance levels with fewer parameters, lower computational costs and less optimisation difficulties. Tan et al. introduced a novel compound method to uniformly scale network architectures, developing the so-called Efficient Networks (EfficientNets) [32]. Howard et al. proposed a new generation of Neural Networks specifically designed for their efficiency, i.e. computational costs. They were thought to work on mobile devices and were thus called Mobile Networks (MobileNets V3) [33]. Finally, several researches tried to deal with large images and high-resolution images, that naturally lead to computationally expensive CNN networks. Patch-based CNNs were introduced by analysing patches of large images [8, 26].

CNNs, with all their architectures, were applied to several classification tasks based on pictorial input data: object classification, street view house number recognition [31], industrial inspection and surface defect detection [4, 5], face detection and image recolouring [34], and traffic sign detection and recognition [8] are just a few examples.

Unlike state-of-the-art CNN architectures, typically featuring wide and deep structures, in this paper we introduce a shallow architecture made up of 3 stacked convolutional blocks. This choice was made to respond to common challenges of deep learning approaches, i.e. to rely on large and exhaustive training datasets (typically featuring more than 50,000 images). The CNN is thought to learn from patches of images in order to limit the number of parameters and reduce architecture complexity when dealing with high-resolution images. Moreover, a CNN was never used as a means to infer the correctness of cutting conditions, in terms of process parameters, from images of the machined surfaces. In the future this will open up the possibility of retrieving process parameter corrective actions from the CNN for adaptive control strategy purposes.

3 Materials

This section presents the milling parameters of interest for this paper and how the ability of producing many specimens influenced the overall classifier development process in terms of the data pre-processing and evaluation metric.

3.1 Experiments

The research activity focused on developing models that are able to recognise and classify different process and technological parameters for milling operations by leveraging only images of machined surfaces. The parameters of interest were:

- Process parameters: machining conditions, feed rate and cutting speed;
- Technological parameters: tool diameter and insert nose radius.

An experimental campaign was designed and carried out in order to produce the specimens required to collect the images to be analysed using the classification models. The specific values of the parameters of interest for the purpose of this paper are shown in Table 1. To reduce the required number of specimens, machining costs and times, it was decided to mill up to 2 sides of the prismatic specimens,

Table 1 Parameters of interest for which the experimental campaign was carried out

Parameter type	Parameter	Values
process	machining conditions	nominal, run-out, chipped tooth
	feed rate [mm/tooth]	0.10, 0.15, 0.20, 0.60, 0.70, 0.80
	cutting speed [m/min]	80, 150, 220
technological	tool diameter [mm]	27.9, 32.0
	nose radius [mm]	0.4, 0.8, 1.0, 1.2, 1.5

and to machine different portions of the same surface with different combinations of cutting parameters (Fig. 1).

The starting dataset consisted of 100 very high-resolution (on average about 8 megapixels) RGB images with a portrait aspect ratio, similar to those shown in Fig. 1 and in Fig. 2 (left). The images were collected using a Keyence VHX-7000 digital microscope focused on a machined area of the specimen (using 20x magnification). Different combinations of milling parameters lead to different patterns on the surfaces (Fig. 1), and some of these patterns can be recognised with ease, while some cannot. The first objective of the classifiers was to detect if surface quality is acceptable or not. A surface was declared unacceptable based on three main aspects: average rugosity R_a higher than $1\mu m$ for finishing operations; too low shininess (related to low cutting speeds and, consequently, built-up edge) and scratched surface (related to chipped tools and excessive run-out). This led to 51 unacceptable surfaces and 49 acceptable ones. Then the classifiers perform a reverse engineering task, in order to correctly recognise technological and process parameters by analysing the input images. The reconstruction of both technological and process parameters used to produce the specific surface allows to infer which parameter caused the surface to be unacceptable.

3.2 Data pre-processing and dataset preparation

The images collected featured very high resolution (about 8 megapixels). Independently of the classification approach, dealing with a high number of pixels would be cumbersome and, more importantly, would imply excessive computational costs. For context, state-of-the-art classification models were trained using image sizes ranging from 32x32 [4, 5, 16, 30, 31] to 600x600 pixels [32], with one of the most commonly adopted being 224x224 pixels [13, 16, 23, 30–33]. Furthermore, due to experimental costs related to machining, the resulting dataset was quite small, and

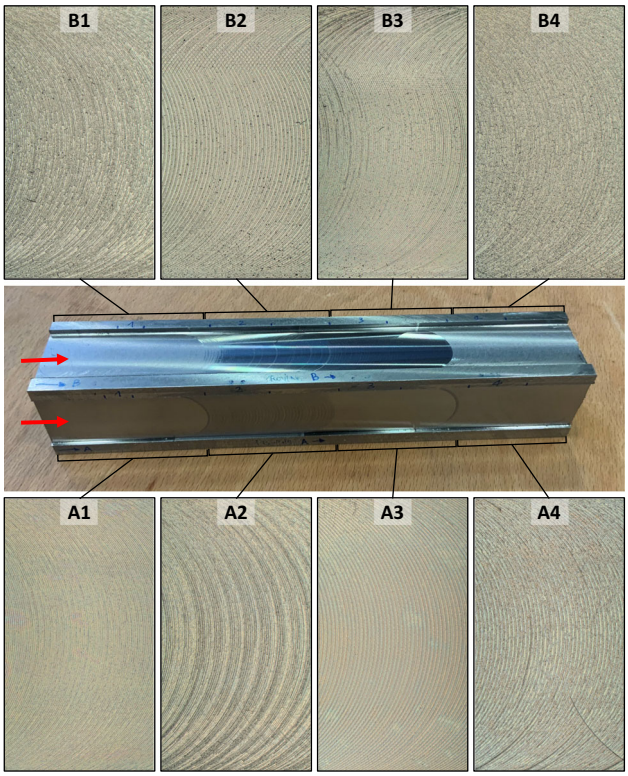

Fig. 1 Example of a specimen used for the milling tests. Red arrows show machined sides, divided in portions with different combinations of cutting parameters indicated by square brackets. Milled surface images samples are reported as example (all with tools in *Nominal* conditions and *Largest* tool diameter). Samples with label *A* were obtained with the *Lowest* nose radius, while *B* ones with a *High* nose radius. A1: feed - *VVL*, cutting speed - *H*; A2: feed - *VVL*, cutting speed - *L*; A3: feed - *L*, cutting speed - *H*; A4: feed - *L*, cutting speed - *L*; B1: feed - *L*, cutting speed - *L*; B2: feed - *VVL*, cutting speed - *H*; B3: feed - *L*, cutting speed - *H*; B4: feed - *VVL*, cutting speed - *L* (where *VVL* is *Very very low*, *L* is *Low* and *H* is *High*)

consisted of 100 total samples. For reference, to train state-of-the-art classification neural networks, researchers leveraged datasets such as Imagenet [35], CIFAR-10 [36] and MNIST [37] consisting of 14 million, 60,000 and 70,000 images, respectively.

To overcome these limitations, it was decided to split each raw image into smaller patches [8, 19, 20, 26] with a dimension of 224x224 pixels. Thanks to this operation, it was possible to:

- Reduce the dimensions of the input images, dramatically decreasing the computational costs required to develop and evaluate the models.
- Increase the number of samples in the dataset, since each raw image got split in about 150 patches.
- deal with overlapping passes and milling strategies with radial engagements different from 100% without the need for dedicated training samples. This is possible

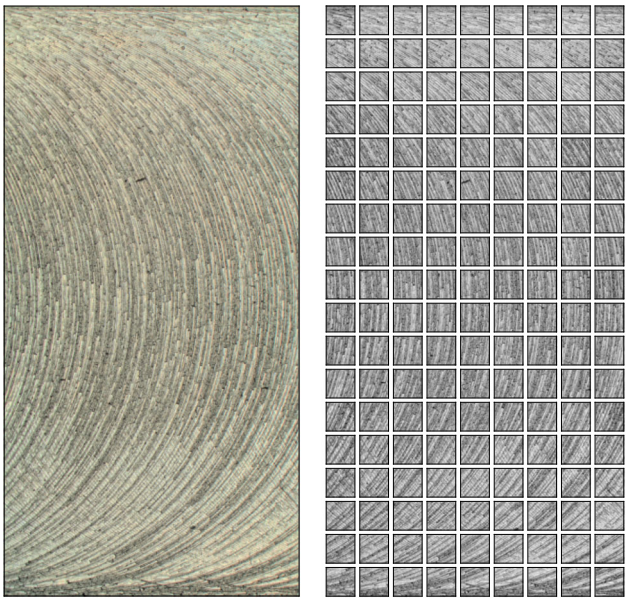

Fig. 2 Left: raw image of a milled surface sample. Right: pre-processing of the same sample image of milled surface: patches of 224x224 pixels are extracted from the image, then converted to grayscale. If this sample image is selected for training purposes, then all of its patches are assigned to the training set

since patches can be drawn from small regions of the milled surface, where no overlap occurs.

With that being said, it must be highlighted that patches from the same image should not be treated as completely distinct samples: as visible from Fig. 2, two neighbouring patches are likely to be characterised by similar patterns. If one of those patches would be used to train the classifier but the other one to test it, their similarity could be a source of information leakage, i.e. leading to an overestimation of the model's prediction capabilities and hiding the model's actual prediction performance. Therefore, to avoid any information leakage between the train and test partitions, it was decided to proceed as follows:

1. For each parameter of interest, separate the 100 available raw images into train and test partitions.
2. Split each raw image into as many 224x224 pixel patches as possible.

Consequently, the image pre-processing pipeline is completed by the following steps, which are commonly adopted in literature [12]:

1. Convert the images' colour space from RGB to grayscale.
2. Scale the pixels values of each sample from the range [0,255] to the range [0,1];
3. Compute pixels mean value across the training fold ($\overline{x}_{tr}$).
4. Centre each training and test sample by subtracting the mean pixel value of the training partition. Thus, subtract the training mean $\overline{x}_{tr}$ from each training and test sample $\mathbf{x}_{i,tr}$ and $\mathbf{x}_{j,te}$ (where $\mathbf{x}_{i,tr}$ and $\mathbf{x}_{j,te}$ are vectors containing all the pixel values of training sample i and test sample j, respectively).

A factor that can negatively influence the performance of the classifiers is the different percentage of samples within each class, which leads to unbalanced datasets. For instance, in the case of the machining condition parameter, collecting data from undesired machining conditions is not trivial and can be expensive as well as risky. Thus, of the 100 available samples, a large portion belonged to *nominal conditions*, whereas only a limited number of samples to *chipped tooth* and *run-out*. Having fewer samples for a given class means that the classifier will struggle to learn relevant features for that class, hindering its overall performance. To compensate for this limitation the classifiers were developed leveraging a 5-fold cross-validation process. This process consisted of evaluating a classifier by looking at a metric averaged over 5 different train-test splits. At each split, a different set of 20% of the available samples was reserved for testing. Furthermore, since some target parameters (both process and technological) presented classes populated by a very limited number of samples (fewer than 10), a stratified split approach was used to preserve the percentage of samples for each class in each train-test split.

3.3 Evaluation metric

Because of its simplicity and interpretability, *accuracy* is the most widely used classification metric, and it is defined as:

$$accuracy = \frac{correctpredictions}{totalpredictions} \tag{1}$$

However, this metric is not suitable for unbalanced datasets. For instance, if a model was to be trained with a dataset consisting of 99 images collected from *nominal machining conditions* and 1 image collected from *run-out*, it could simply learn to assign the 100 samples to the first class and still score 99% accuracy. Once deployed for production, it would still assign any new sample to the first class, since it did not learn how to distinguish between different machining conditions.

For this reason, a different classification metric was chosen to better quantify the performance of the classifiers when dealing with unbalanced datasets: the f-score. This metric is defined in [38] as the harmonic mean of precision and recall:

$$F_\beta = \frac{(\beta^2 + 1)(P \cdot R)}{\beta^2(P + R)} \tag{2}$$

where:

- β is a parameter that controls a balance between precision and recall;
- P is precision, defined as $\frac{true\ positive}{true\ positive + false\ positive}$;
- R is recall, defined as $\frac{true\ positive}{true\ positive + false\ negative}$.

When the same weight is assigned to precision and recall, meaning $\beta = 1$, the f-score becomes what is commonly referred to as f1-score [39]:

$$F_1 = \frac{2\,P\,R}{P + R} \tag{3}$$

Considering the previous example, the classifier returns an f1-score equal to 0, highlighting the fact that it actually did not learn anything useful and can't be deployed for production.

In Section 5, the classifiers performance will be evaluated, for the 5-folds, in terms of mean value and standard deviation of testing f1-score.

4 Methods

This section presents the traditional texture descriptors, namely LBP, DCP, HOG and GLCM, that were used to train the machine learning models (SVM, KNN, RF) to classify the milled surfaces images according to the parameters of interest (explained in Section 3.1). The results obtained using these approaches constitute the benchmarks for the performance of the deep learning approach arrived at. As a final remark, it is necessary to remind readers that the data pre-processing steps implemented before computing the texture descriptors are the same as those presented in Section 3.2, and the evaluation metric is as presented in Section 3.3.

4.1 Theoretical background

4.1.1 Traditional texture descriptors: LBP, DCP, HOG, GLCM

From a general point of view, the common starting point for each traditional texture descriptor are the pre-processed images. Once the descriptors are computed, it is possible to proceed in two ways:

1. Use the resulting images as inputs to the ML models.
2. Perform a further step and compute features from the resulting images, and use those features as input to the ML models.

Because the ML models implemented in this paper are not compatible with images (images in the form of 2D arrays), it was decided to implement option 2. It could be argued that the images can be converted into a stack of 1D arrays, but this implies working with very large inputs, that dramatically increase the computational costs. Furthermore, it would lead to a situation in which the number of features vastly exceeds the number of samples, commonly referred to as the *curse of dimensionality* [40]. Consequently, once the LBP, DCP, HOG and GLCM descriptors were computed for each starting sample, a further reduction step was added to each descriptor (e.g. histogram extraction, principal component analysis), obtaining a set of limited feature vectors. This reduction step was specific for each texture desciptor. Indeed it will be explained, together with the associated descriptor in the following.

LBP computes a histogram with the distribution of the binary configurations of the pixels of the image, based on thresholding the surrounding window of each pixel with the intensity of the centre pixel. Generally, the LBP descriptor works on 3x3 pixels cells, and the centre pixel's value sets the threshold. Each neighbouring pixel is converted into a binary value according to this criterion:

- If the neighbouring pixel's value is larger than the threshold, the pixel is converted to 1.
- If the neighbouring pixel's value is smaller than the threshold, the pixel is converted to 0.

The resulting sequence of zeros and ones is converted to a binary number and thus to an integer number (according to the binary conversion system). This value is assigned to the centre pixel of the starting 3x3 cell; the same procedure is repeated for all the other pixels in the image to obtain an output image. Figure 3 clarifies how a sample 500x500 pixels window changes when LBP is computed, while Fig. 4 relates to a 100x100 pixels window from the same original sample. A histogram of the output image was computed and its histogram bin values were selected as the feature vector. The idea is that surfaces that show different patterns should present different LBP histograms, and thus, the bins should have different values. Indeed, it should be possible to train ML classifiers to recognise those differences.

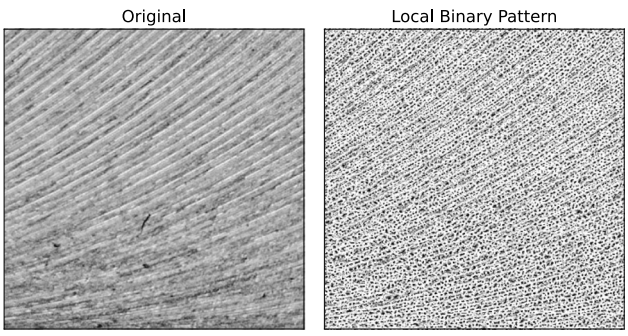

Fig. 3 Left: 500x500 pixels bottom-left corner of a sample image. Right: 500x500 pixels of the same portion plotted in terms of LBP values

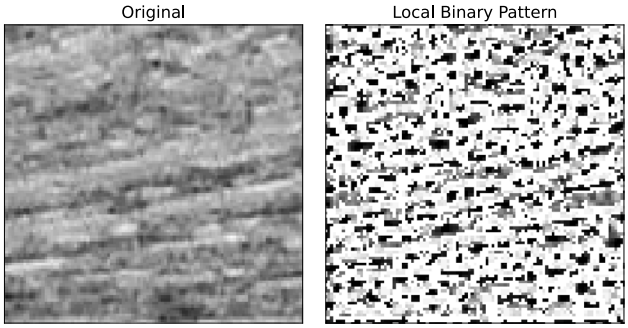

Fig. 4 Left: 100x100 pixels bottom-left corner of a sample image. Right: 100x100 pixels of the same portion plotted in terms of LBP values

DCP allows to extract two different pixel maps, encoding the texture patterns of machined surfaces. DCP works on a two-level scale defined by an internal and external radius (R_{in} and R_{ext}, respectively). Here, $R_{in} = 4$ and $R_{ext} = 6$, following [21]. Eight pixel values are drawn on the internal circle and eight on the external circle from a central pixel, with angle steps of $\pi/4$ radians. A decimal number is assigned to each direction through a two-step difference (i.e. difference from inner radius to centre and difference from external radius to inner radius). This number is then binarised with a threshold at the zero level. The binarised pixel values are then grouped into two crosses (from which the descriptor takes its name) and encoded into a pixel value assigned to the central pixel. The procedure is repeated for each pixel in the original image, with each cross generating a processed image. Figures 5 and 6 show the two images produced by the two DCP crosses for a 500x500 sample and a 100x100 sample, respectively. Histograms of these images are computed and concatenated in the texture descriptor itself. More details on DCP computation can be found in [15, 21].

HOG descriptors are mainly used to describe the structural shape and appearance of an object. However, since HOG is able to capture the local intensity gradients and edge directions, it is also a good texture descriptor. By focusing on the gradient and orientation of the edges (magnitude and direction), it is possible to understand whether the pixels belong to an edge and find its orientation. The idea is to leverage the edges and their orientations to distinguish between different cutting parameters that generate different patterns on the metallic surfaces. The orientations were calculated in localised portions, meaning that the starting image was decomposed into smaller regions (or *cells*) and, for each region, the gradients and orientations were calculated. Consequently, a histogram for each region was generated from the gradients and orientations of the pixel values. An example of the output from the HOG descriptor is shown in Fig. 7 for a window of size 500x500 pixels in size and Fig. 8 for a window of size 100x100 pixels in size. Moreover, from Fig. 8 it is possible to see how the starting image was broken down into smaller regions (with dimensions 32x32 pixels) and a total of 9 possible orientations was set. To limit the issues caused by the very high dimensionality implied by the adoption of these texture descriptors, it was decided to pass the HOG values through a principal component analysis (PCA) [41] stage to obtain a feature vector with a length of 8 for each sample.

GLCM consists of computing statistical experiments on the matrix (or matrices) that contain the co-occurrences of the pixel intensities at given angles and distances. These statistical experiments intuitively provide measurements of properties such as smoothness, coarseness and regularity on the distribution of pixels within the texture. By definition a GLCM is the probability of the joint occurrence of grey-levels i and j, where $i \leq G$ and $j \leq G$ and G identifies the grey-level depth of the image, within a defined spatial relation in an image. This spatial relation is defined in terms of a distance D and an angle θ. From the GLCM, it is possible to compute statistical features that can then be stacked together to build the feature vector. In particular, the features evaluated according to [42] were:

- contrast;
- dissimilarity;
- homogeneity;
- energy;
- correlation.

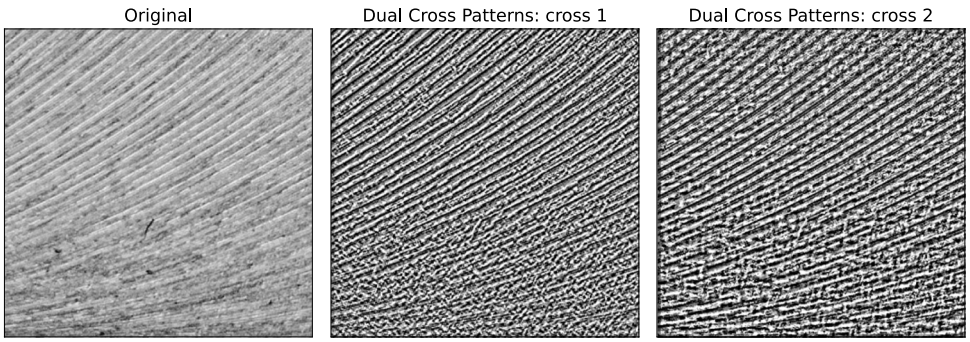

Fig. 5 Left: 500x500 pixels of bottom-left corner of a sample image. Centre: 500x500 pixels of the same portion plotted in terms of DCP values with cross at $[0, \pi/2, \pi, 3\pi/2]$ directions. Right: 500x500 pixels of the same portion plotted in terms of DCP values with cross at $[\pi/4, 3\pi/4, 5\pi/4, 7\pi/4]$ directions

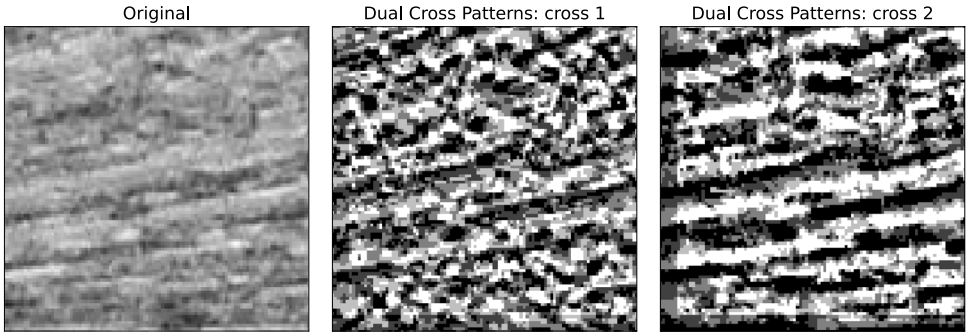

Fig. 6 Left: 100x100 pixels of bottom-left corner of a sample image. Centre: 100x100 pixels of the same portion plotted in terms of DCP values with cross at $[0, \pi/2, \pi, 3\pi/2]$ directions. Right: 100x100 pixels of the same portion plotted in terms of DCP values with cross at $[\pi/4, 3\pi/4, 5\pi/4, 7\pi/4]$ directions

In this paper, to compute the 5 features it was decided to set D equal to 100 and θ equal to 0°. These values might not be the best ones, and evaluating the best possible combination is actually very time consuming since a sweep of all possible combinations would be required. This is one of the disadvantages of trying to classify images when leveraging highly engineered features such as GLCMs.

An example of the GLCM evaluated is shown in Fig. 9 for two different cutting speed values. It is possible to notice that the resulting GLCMs were slightly different, and these matrices were leveraged to compute the statistical parameters mentioned above. In this example, the difference between the two raw samples was highlighted by the *contrast* feature, while the other features didn't highlight large differences.

4.1.2 Machine learning classifiers: SVM, KNN, RF

To establish a performance baseline for the classification problem presented in this paper, it was decided to utilise three of the most widely known supervised machine learning classifiers, namely SVM, KNN and RF.

Considering a binary classification problem, meaning classifying between just two classes, the objective of the SVM was to find the hyperplane that maximises the distance to the nearest input data point of any class, as per Fig. 10 (the distance is called margin, the nearest sample is called support). SVMs proved to be able to perform both linear and non-linear classification; for the latter, a *kernel trick* [43] is exploited to map the input data onto a higher dimensional space. The idea behind the *kernel trick* is that if the input data is not linearly separable in the current space, it may be separable at a higher dimension, as shown in Fig. 11, computed using combinations of simple functions and the starting input data. Two of the best known kernel functions are polynomial and gaussian radial basis functions. The disadvantage of the *kernel trick* is that by increasing the dimensionality of the problem, the testing error increases, too; thus, to compensate for this, more data is required to train the SVM.

The KNN classification algorithm [24] is quite intuitive: for any new sample, a distance metric is computed to find the k-nearest samples and assign it to a class according to a voting mechanism based on the most recurring class among the k-neighbours (Fig. 12). The main parameter that has to be tuned is k, which sets the number of neighbours considered to classify a new sample (i.e. participants in the voting mechanism):

– If k is too high the testing error increases because the boundaries between classes are less distinct. This means that samples of different classes can be mixed together and assigned to a single region (underfitting). When a new sample falls within that boundary, it may be

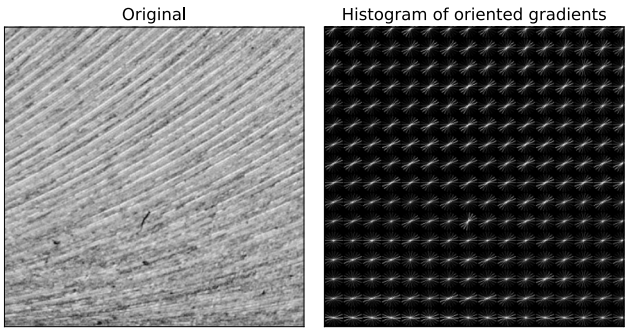

Fig. 7 Left: 500x500 pixels bottom-left corner of a sample image. Right: 500x500 pixels of the same portion plotted in terms of HOG

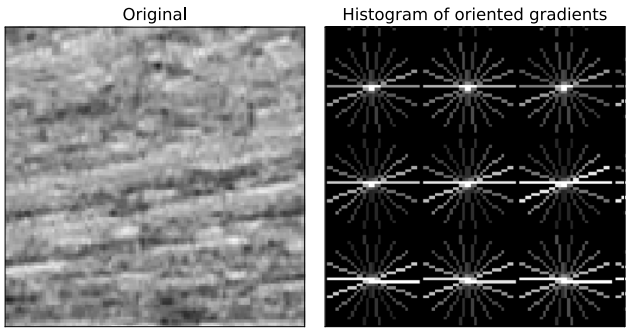

Fig. 8 Left: 100x100 pixels bottom-left corner of a sample image. Right: 100x100 pixels of the same portion plotted in terms of HOG

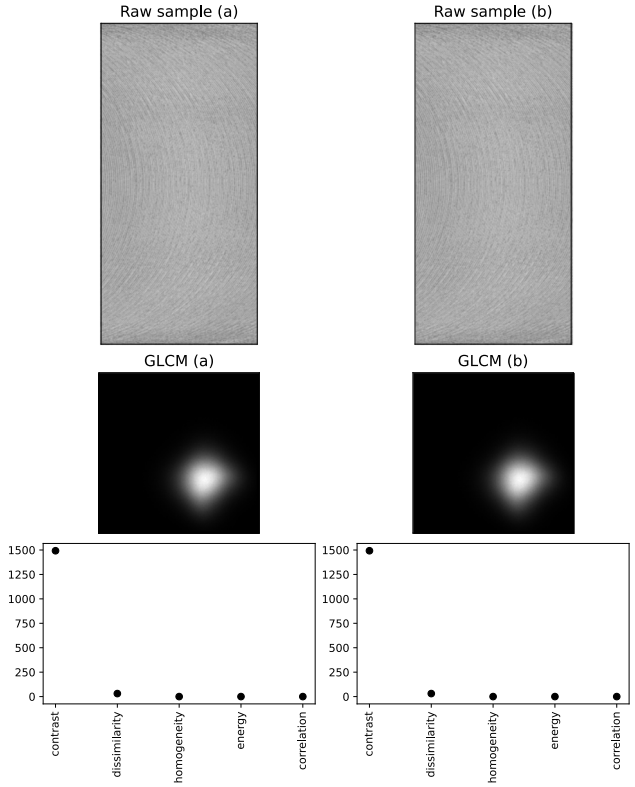

Fig. 9 Top: two raw samples representing different cutting speed values (a: high cutting speed, b: low cutting speed). Middle: GLCM from raw samples (a) and (b). Bottom: features computed from GLCM (a) and (b) respectively

incorrectly classified because the most recurring class among its k-neighbours is actually different from that for the region (which is the correct one).

– If k is too low the computational cost and time increase dramatically, especially if the dataset is large and the samples have high dimensionality. Furthermore, KNN may end up learning the noise (overfitting).

So, k is usually tuned by applying some optimisation methods that test different possible values and evaluate the performance of the KNN classifier accordingly.

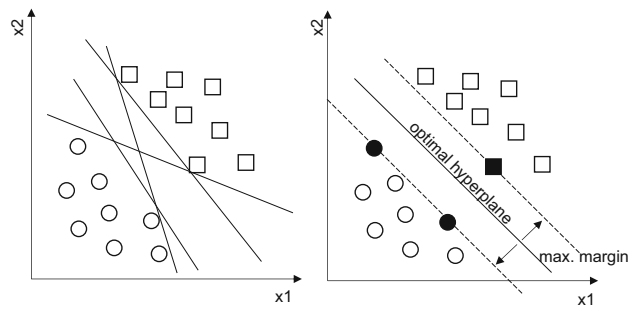

Fig. 10 SVM defines the hyperplane that maximises the margin between the samples of the two classes

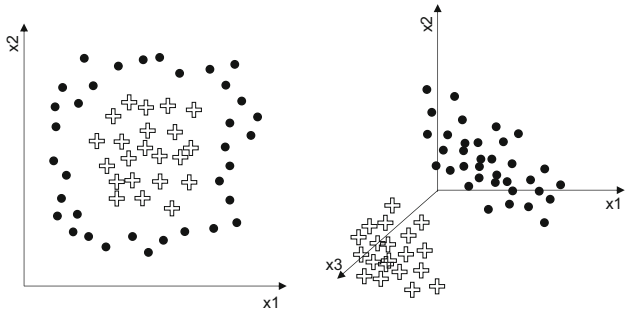

Fig. 11 SVM: example of kernel trick application and how it makes it possible to define a plane that separates the two classes in a higher dimensional space

Random forest classifiers are built by combining multiple decision trees that operate as an ensemble [28]. The main features that characterise the random forest classifiers are:

– Bootstrap aggregating algorithm (bagging), leveraged to assign the input data to each decision tree.
– Each decision tree operates on a random subset of features to reduce the correlation between different trees.
– The classification output of the random forest is selected based on the most voted class by the decision trees.

To better adapt the random forest classifier to the specific task it is possible to tune a variety of parameters, such as: the number of estimators (decision trees), the number of features for each estimator and the maximum depth of

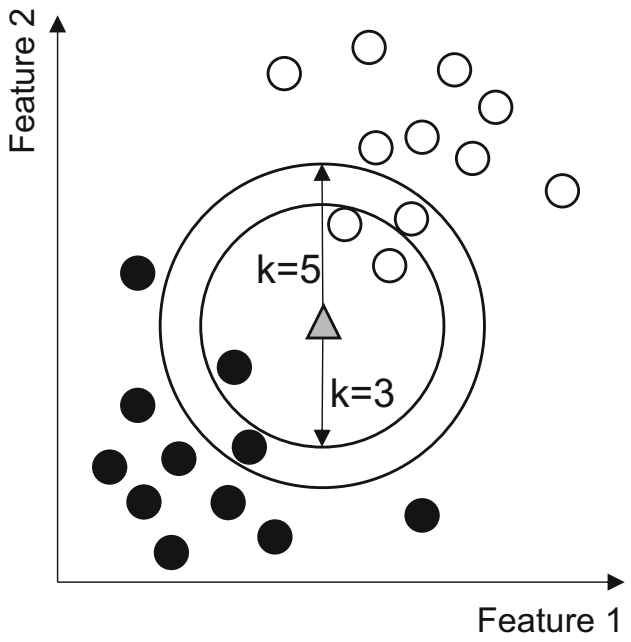

Fig. 12 KNN: example of classification with a different number of neighbours

each estimator. The number of estimators sets the number of decision trees that will be used to build the random forest classifier. It is not always clear if this parameter should be tuned or set to the maximum value possible given the available computing resources [44]. The number of features for each estimator sets the number of features that will be leveraged by each decision tree to classify any given sample, and it is usually set equal to the square root of the number of total features available. The depth of the estimators can be limited to a maximum value to avoid overfitting the training data and to improve the bias-variance trade-off (very deep decision trees lead to very good performance at the expense of becoming very sensitive to small variations in the input data, which leads to large variations in the predicted class).

From an operative perspective, the three algorithms (SVM, KNN, RF) were tuned and trained using a common strategy that leveraged a combination of cross-validation and grid-search. As reported in the pseudo-code in Algorithm 1, the first step was loading the data (meaning the raw images) and computing a set of traditional feature descriptors such as LBP, DCP, HOG or GLCM. The different sets of tuneable hyperparameters characterising each model in this paper were the following:

- SVM: C, kernel, polynomial kernel degree.
- KNN: number of neighbours, weights, power parameter for Minkowski distance.
- RF: number of estimators, maximum number of features.

For each hyperparameter multiple possible values were set. The second *for* loop, split the dataset into an 80%-20% training-testing proportion. In the third *for* loop, the model was tuned and trained. Here, two approaches were deployed at the same time: grid-search was used to generate all possible hyperparameters' value combinations, 5-folds cross-validation was used to evaluate which combination led to the best testing results. The third *for* loop, was actually repeated 20 times (see line 7 in Algorithm 1): the objective was to find the most recurring best architecture from the grid-search cross-validation process to limit the detrimental effects of imbalanced datasets as much as possible.

```
1: set parameters
2: set models
3: for parameter in parameters do
4:     load data
5:     compute feature (LBP, DCP, HOG, GLCM)
6:     set hyperparameters grid boundaries
7:     for i in range(20) do
8:         stratified train-test split of data (80-20)
9:         initialize Grid-Search as gs
10:        initialize StratifiedShuffleSplit as sss
11:        initialize Cross-Validation as cv
12:        for model in models do
13:            initialize model
14:            train and tune model with gs, sss and cv
15:            compute F1score
16:            save tuned model architecture
17:        end for
18:    end for
19:    find most recurring model architecture
20: end for
```

Algorithm 1 SVM, KNN, RF train-test loop.

4.2 Custom CNN for surface classification

This section explains the proposed CNN model's architecture, highlighting some of the design choices that were made to compensate for the limitations introduced in Section 3.2.

It is possible to notice from the graph in Fig. 13 that the proposed CNN was very shallow and consisted of a series of stacked convolutional blocks. This went against the design trends of deeper (and sometimes wider) models that characterised recent state-of-the-art convolutional neural networks for image classification such as ResNets [30], DenseNets [31] and EfficientNets [32], just to name a few.

Keeping the network shallow and limiting the number of trainable parameters was necessary due to the dataset shortcomings highlighted in Section 3.2:

- Training deep models, with many trainable parameters, leveraging only a limited number of samples would inevitably lead to overfitting. This overfit tendency would hinder the model's performance when deployed for production.
- Having a model with few trainable parameters facilitates the training and inference processes when dealing with limited computing resources.

Nonetheless, if necessary, it is possible to increase the depth of the proposed model by stacking more convolutional blocks. Similarly, the model can be widened by increasing the number of filters generated by each convolution operation.

From an operating perspective, the input images were fed into a first convolutional block (*stem block*), with the purpose of increasing the number of channels while decreasing the spatial dimensions. The stem was very similar to implementations found in literature, consisting of a convolutional layer with a large filter size followed by the batch normalisation, rectified linear unit (relu) activation function and a max pool operation for further down-sampling. The resulting outputs were passed through a stack of three inverted residual linear blocks (called *conv block*,

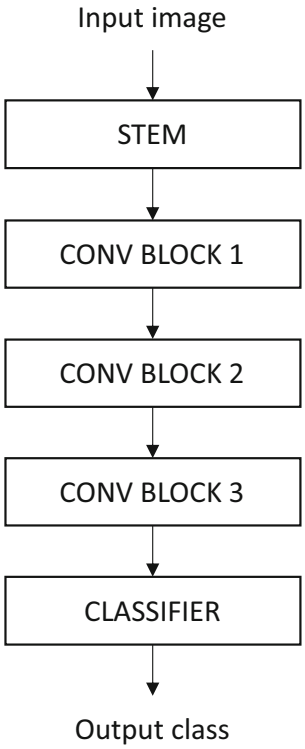

Fig. 13 Compact CNN architecture overview

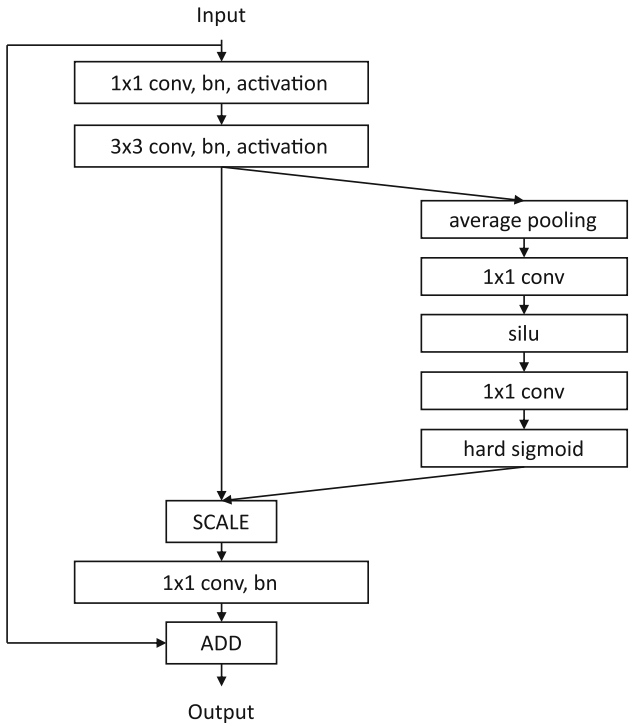

Fig. 14 Convolutional block detailed view

based on MobileNetV3 [33] and EfficientNets) detailed in Fig. 14: the *conv blocks* were characterised by state-of-the-art features such as depth-wise convolutions, squeeze-excite blocks, residual connections and hard-sigmoid activation function. The inverted residual linear block was integrated into the proposed CNN and custom developed to:

- Be lightweight.
- Optimise the accuracy-latency trade-off on limited resources (mobile devices), which are desirable features for this paper's research purposes.
- Efficiently generate feature maps that synthesise meaningful information from each sample, making it possible to recognise the different machining parameters of interest.

The last stage of the proposed CNN was the *classifier block*, where the output filters were averaged and passed through a convolutional layer and softmax layer combination in order to obtain the predicted classifications.

A detailed summary of all the parameters that defined the architecture of the proposed CNN (dimensions, kernel sizes, filters, strides, activation functions) is given in Table 2. Some further details regard the squeeze-excite block, which adopted convolutional layers as weight layers and it was characterised by the absence of batch-normalisation layers. The reduction factor was set to 8. In the *classifier block* the dropout layer had a survival probability of 90%.

The proposed CNN model was trained using the stochastic gradient descent algorithm (SGD) with an initial learning rate 0.9 and momentum 0.9. The learning rate was adjusted after every batch according to the *OneCycle* policy [45]. The number of epochs was set to 20 and batch size to 128.

5 Results

In this section the classification results obtained from the two approaches are reported, that is:

- Traditional texture descriptors (LBP, DCP, HOG, GLCM) and machine learning classifiers (SVM, KNN, RF);
- End-to-end classifier model (CNN).

Firstly, the acceptability of the surface is classified, then for each process (machining condition, feed rate, cutting speed) and technological (tool diameter, tool nose radius) parameter, the classifiers' performance was evaluated via the f1-scores over the 5-fold cross-validation process. The results in the following tables describe the classification performance levels in terms of testing f1-score mean value and its standard deviation.

Table 2 Custom CNN architecture

Output size	Layers	Activation
54x54	conv, 11x11, 32, stride=4	relu
26x26	max pool, 3x3, stride=2	
13x13	$\begin{bmatrix} conv, 1x1, 64 \\ dwconv, 3x3, 64, stride = 2 \\ conv, 1x1, 48 \end{bmatrix}$ x1	relu
7x7	$\begin{bmatrix} conv, 1x1, 96 \\ dwconv, 3x3, 96, stride = 2 \\ conv, 1x1, 64 \end{bmatrix}$ x1	hard-sigmoid
7x7	$\begin{bmatrix} conv, 1x1, 128 \\ dwconv, 3x3, 128, stride = 2 \\ conv, 1x1, 64 \end{bmatrix}$ x1	hard-sigmoid
1x1	global average pool dropout	
1x1	conv, 1x1, classes	softmax

5.1 Surface acceptability classification results

Table 3 shows the testing performance levels of each model when classifying the surface samples as:

- Acceptable.
- Unacceptable.

The proposed CNN was the best performing model when classifying the acceptability of a surface. The CNN marginally outscored the RF+HOG combination (0.91801 and 0.91667, respectively). All the other state-of-the-art combinations led to lower f1-scores, ranging from 0.61667 to 0.88750. The CNN showed a higher reliability in the classes prediction, which was highlighted by lower standard deviation of the f1-score. In fact, CNN standard deviation was ±0.02953, which is about three times lower with respect to the RF+HOG combination (±0.08756). This may be associated to the capability of the deep learning algorithm to extract meaningful features compared to traditional descriptors. In general, the recognition of an unacceptable surface seem to be an easy task, aided by the fact that it is a binary classification problem. Machine learning algorithms seem to have a low influence on the classification results, which are mainly driven by the feature extraction process. GLCM was the worst descriptor for this case, performing badly with any machine learning algorithm (0.61667 is the lowest f1-score, obtained with KNN, whereas 0.71364 is the highest one when paired with SVM). DCP slightly outscored GLCM, still resulting ineffective for the classification of the surface quality. Despite HOG and LBP resulted to be good traditional surface quality descriptors, they did not allow to reach the CNN results.

5.2 Machining condition classification results

Table 4 shows the testing performance of each model when classifying the samples according to the machining conditions. We wish to remind that the available samples in the dataset were collected for 3 machining conditions:

- Nominal.
- Run-out.
- Chipped tooth (insert).

When classifying the three machining conditions, the best performing model was the proposed end-to-end CNN, which marginally outscored the KNN+GLCM and RF+LBP combinations (0.86838 compared to 0.86667 and 0.86364). It is interesting to note how the CNN showed slightly more reliable testing performance, highlighted by a lower standard deviation than KNN+GLCM (±0.02786 vs ±0.03275). This could be an indicator of the CNN's better generalisation capabilities compared to KNN+GLCM, and of its effectiveness in learning synthetic features that capture meaningful information within each sample. With that being said, the classifier and feature pairs were able to perform very well on average, with f1-scores comfortably above the 80% threshold. Unexpected behaviour was shown by SVM when paired with LBP, DCP and GLCM: SVM was completely unable to properly separate the samples when leveraging features obtained from these three texture descriptors, and this was true across the different parameters analysed (see tables in the pages that follow). Considering that the SVM's hyperparameters were tuned using a grid-search approach and that the samples were cycled by cross-validation, it is fair to assume that this phenomenon is due

Table 3 Surface acceptability classification results

ML algorithm	Texture descriptor	F1-score (5-fold)
SVM	LBP	0.82000 ± 0.07888
	DCP	0.72500 ± 0.11292
	HOG	0.86818 ± 0.07833
	GLCM	0.71364 ± 0.07103
KNN	LBP	0.85000 ± 0.07906
	DCP	0.70000 ± 0.05477
	HOG	0.88750 ± 0.06440
	GLCM	0.61667 ± 0.07906
RF	LBP	0.84167 ± 0.07360
	DCP	0.69286 ± 0.07868
	HOG	0.91667 ± 0.08756
	GLCM	0.66875 ± 0.06512
CNN		**0.91801±0.02953**

Bold line represent the algorithm with the best f1-score

Table 4 Machining condition classification results

ML algorithm	Texture descriptor	F1-score (5-fold)
SVM	LBP	0.10000 ± 0.00000
	DCP	0.10000 ± 0.00000
	HOG	0.83750 ± 0.03536
	GLCM	0.10000 ± 0.00000
KNN	LBP	0.80714 ± 0.07319
	DCP	0.86250 ± 0.03536
	HOG	0.84615 ± 0.04770
	GLCM	0.86667 ± 0.03257
RF	LBP	0.86364 ± 0.02335
	DCP	0.85556 ± 0.01667
	HOG	0.85000 ± 0.00000
	GLCM	0.83000 ± 0.02582
CNN		**0.86838±0.02786**

Bold line represent the algorithm with the best f1-score

to a poor choice in the features selection process (namely in terms of type of features and quantity of features). It is also true that LBP, DCP and GLCM features did perform well when leveraged to train other ML models, so this problem appears to be limited to the SVM.

5.3 Feed rate classification results

Table 5 shows the testing performance of each model when classifying the samples according to the feed rate. Remember that the available samples in the dataset were collected for a total of 6 different feed rate classes:

- Very very low.
- Very low.
- Low.
- High.
- Very high.
- Very very high.

For this classification task, the best performing model was the proposed CNN, clearly outscoring all the tested models-texture descriptors combinations by quite some margin (the second best model-texture descriptor combination is RF+DCP, trailing by more than 2 basis points). As noted for the machining conditions classification, the proposed CNN was very stable, registering the second-lowest f1-score spread (±0.01170). With that being said, it is possible to notice how each model was struggling to correctly classify the samples according to the feed rate. The proposed CNN was the only model that achieved meaningful performance levels, with the remaining models registering very low f1-scores (below 50%, except for RF+DCP and SVM+DCP). This is likely due to the models assigning most of the testing samples to just one or two classes (out of the six available), therefore the training process didn't have the expected impact. One other reason could be that recognising this cutting parameter simply by looking at an image is hard even for human experts. This may be amplified for models trained on a limited number of samples.

5.4 Cutting speed classification results

Table 6 shows the testing performance of each model when classifying the samples according to the cutting speed.

Table 5 Feed rate classification results

ML algorithm	Texture descriptor	F1-score (5-fold)
SVM	LBP	0.10000 ± 0.03536
	DCP	0.54167 ± 0.07360
	HOG	0.47500 ± 0.08018
	GLCM	0.10000 ± 0.00000
KNN	LBP	0.42000 ± 0.04472
	DCP	0.41000 ± 0.12450
	HOG	0.50000 ± 0.03162
	GLCM	0.38333 ± 0.06124
RF	LBP	0.44375 ± 0.10155
	DCP	0.57500 ± 0.08216
	HOG	0.47000 ± 0.07583
	GLCM	0.33889 ± 0.09610
CNN		**0.59577±0.01170**

Bold line represent the algorithm with the best f1-score

Table 6 Cutting speed classification results

ML algorithm	Texture descriptor	F1-score (5-fold)
SVM	LBP	0.67692 ± 0.05250
	DCP	0.66176 ± 0.10082
	HOG	0.81667 ± 0.11902
	GLCM	0.51667 ± 0.17224
KNN	LBP	0.82143 ± 0.05669
	DCP	0.62500 ± 0.15083
	HOG	0.76111 ± 0.12693
	GLCM	0.61429 ± 0.10690
RF	LBP	0.78000 ± 0.05701
	DCP	0.73000 ± 0.10368
	HOG	0.78571 ± 0.06901
	GLCM	0.66875 ± 0.05939
CNN		**0.88718±0.03529**

Bold line represent the algorithm with the best f1-score

Remember that the available samples in the dataset were collected for a total of 3 different cutting speed classes:

- Low.
- Mid.
- High.

Similarly to the previous classification tasks, the best performing approach for the cutting speed classification was the proposed CNN, registering a mean f1-score of 0.88718 over the 5-folds. The second best performing approach was represented by the KNN+LBP combination with a mean f1-score of 0.82143. Once again, the proposed CNN showed very stable classification performance, highlighted by a testing f1-score standard deviation of ±0.03529 which is the tightest across all other models-feature combinations. For reference, the KNN+LBP combination shows a f1-score spread that is almost 60% wider (±0.05669) than that of the proposed CNN. The overall testing performance of all models was acceptable, with most model-feature combinations registering f1-scores above the 70% threshold, signalling that it is feasible to recognise different values of the cutting speed parameter.

5.5 Tool diameter classification results

Table 7 shows the testing performance of each model when classifying the samples according to the tool diameter. Remember that the available samples in the dataset were collected for a total of 2 different tool diameter classes:

- Small.
- Large.

For this classification task it is possible to see that almost all the approaches performed very well, registering testing f1-scores above the 90% mark. With that being said, the benchmark was set by the SVM+HOG combination, showing a mean test f1-score of 0.98636 which slightly outperformed the proposed CNN model (0.97464). At the same time, the proposed CNN showed excellent performance reliability across the 5-folds: it registered a test f1-score deviation of just ±0.00785, which was a third of the deviation registered by the SVM+HOG combination. This confirmed both the effectiveness of the features generation and learning processes and the generalisation capabilities of the proposed CNN model, even when classifying a technological parameter such as the tool diameter. One must remember that this classification task was somewhat simpler compared to the other tasks, since it only had to distinguish between just 2 classes. Consequently, since the total number of samples was fixed and equal to 100, more samples were available to describe each class, which is desirable to effectively train machine learning models.

5.6 Nose radius classification results

Table 8 shows the testing performance of each model when classifying the samples according to the insert nose radius. Remember that the available samples in the dataset were collected for a total of 5 different nose radius classes:

- Lowest.
- Low.
- Mid.
- High.

Table 7 Milling tool diameter classification results

ML algorithm	Texture descriptor	F1-score (5-fold)
SVM	LBP	0.89333 ± 0.07528
	DCP	0.94167 ± 0.04618
	HOG	**0.98636 ± 0.02335**
	GLCM	0.80000 ± 0.04472
KNN	LBP	0.91000 ± 0.02236
	DCP	0.92500 ± 0.10000
	HOG	0.97727 ± 0.02611
	GLCM	0.82188 ± 0.05468
RF	LBP	0.93333 ± 0.04330
	DCP	0.95000 ± 0.04472
	HOG	0.95714 ± 0.05345
	GLCM	0.76429 ± 0.03780
CNN		0.97464 ± 0.00785

Bold line represent the algorithm with the best f1-score

Table 8 Milling tool insert's nose radius results

ML algorithm	Texture descriptor	F1-score (5-fold)
SVM	LBP	0.50000 ± 0.08165
	DCP	0.59643 ± 0.11679
	HOG	0.76667 ± 0.09374
	GLCM	0.59000 ± 0.09947
KNN	LBP	0.67727 ± 0.10574
	DCP	0.75000 ± 0.06124
	HOG	0.71429 ± 0.08522
	GLCM	0.48571 ± 0.08187
RF	LBP	0.54000 ± 0.15572
	DCP	0.61667 ± 0.08756
	HOG	0.74375 ± 0.08634
	GLCM	0.43125 ± 0.10329
CNN		**0.79475±0.04724**

Bold line represent the algorithm with the best f1-score

– Highest.

The proposed CNN model was the best performing approach when classifying the samples according to the nose radius of the milling tool's inserts. It managed to outperform the SVM+HOG combination, registering a mean test f1-score of 0.79475 compared to 0.76667. As previously shown across the other classification tasks, the proposed CNN was a very reliable approach, with a test f1-score deviation of ±0.04724, which is almost half the score for the SVM+HOG combination (±0.09374). Overall, the results were mixed: HOG features led the machine learning models to the best performance followed by DCP, while LBP and GLCM were unable to generate features that correctly identify the different nose radii from the surface images. It should be noted that, similarly to the feed rate, recognising different values of nose radii through vision only is a complex task even for experts.

5.7 Average classification results

From an overall perspective, it was possible to note that the proposed CNN performed very well across the whole spectrum of classification tasks: it was the best performing model in the case of surface acceptability, machining conditions, feed rate, cutting speed, nose radius classification, and it was marginally outperformed in the case of tool diameter (−1.2% compared to the SVM+HOG combination). This can be seen from Table 9,

Table 9 Average classification performance across the parameters of interest (surface acceptability, machining condition, feed rate, cutting speed, tool diameter, nose radius)

ML algorithm	Texture descriptor	Global F1-score
SVM	LBP	0.51504 ± 0.31808
	DCP	0.59442 ± 0.25476
	HOG	0.79173 ± 0.15673
	GLCM	0.47005 ± 0.27648
KNN	LBP	0.74674 ± 0.16234
	DCP	0.71208 ± 0.16752
	HOG	0.78105 ± 0.15161
	GLCM	0.63142 ± 0.17073
RF	LBP	0.73373 ± 0.17895
	DCP	0.73668 ± 0.13048
	HOG	0.78721 ± 0.15921
	GLCM	0.61699 ± 0.17525
CNN		**0.83979±0.12175**

Bold line represent the algorithm with the best f1-score

where the average performance of the different model-feature combinations and proposed model across the 5 classification tasks are reported:

– The proposed CNN was, on average, the best performing model across all tasks, with an average f1-score of 0.83979. It was the most consistent too, highlighted by a f1-score spread of ±0.12175.
– The second best approach was, on average, the SVM paired with HOG features. This approach showed an average f1-score of 0.79173 (-4.8% compared to the proposed CNN) and a standard deviation of ±0.15673 (+3.5% compared to the proposed CNN).
– The worst approaches were, on average, the SVM paired with GLCM features or LBP features (as stated before, when discussing the machining conditions results).
– From the texture descriptor point of view the most effective approach appears to be the HOG (remember that this feature was paired with PCA) followed by DCP and LBP, while the least effective was the GLCM.

6 Conclusions

In this paper, we have presented a deep learning approach for machined surface classification tasks. A shallow end-to-end Convolutional Neural Network (CNN) classifier was built and trained upon 100 raw surface very high-resolution images, split into 224x224 pixel batches. The CNN learned to classify images for 6 classification tasks: machining conditions, feed rates, cutting speeds, tool diameters and nose radii. The proposed approach was compared to state-of-the-art machine learning techniques (Support Vector Machines, k-Nearest Neighbours and Random Forests), fed with traditional surface feature descriptors (Local Binary Patterns, Dual Cross Patterns, Histogram of Oriented Gradients and Grey-Level Co-Occurrence Matrix). The approach developed outperformed state-of-the-art machine learning techniques in all the classification tasks, except for the milling tool diameter showing a mean f1-score of 97.5% (just 1.2% less than the best state-of-the-art algorithm). Furthermore, the CNN results were 4.8% better than the best machine learning approach when considering the average classification performance levels. Thus, the proposed CNN structure proved to be robust, reliable, flexible and accurate enough for possible industrial deployment. The CNN training procedure on the whole dataset (not considering cross-validation) took 149.7 min, whereas the prediction of a new image class (including all its patches) is performed in 0.385 s, on average. Thus, the CNN is really fast and does not prevent its application in an industrial context (the application time is related just to the prediction time

for a new image). The CNN training and test times were computed on a Dell XPS 15 7590 featuring an Intel® Core™ i7-9750H CPU @ 2.60GHz. Nevertheless, there are still margins for improvement especially regarding the feed rate classification, where the CNN still struggles to reach optimal performance. This will be the subject of the authors' future works, together with the deployment of the approach developed in an adaptive process control framework (i.e. the CNN will suggest milling parameter changes and tool failure detection). Furthermore, when testing the approach on different materials, i.e. Aluminium alloys, the algorithm did not provide reliable results, most likely due to the material reflectivity. Thus, in order to enhance the generalisation properties of the developed CNN and deal with different materials, at least some training samples (performed on the new materials) are needed.

Acknowledgements The authors would like to thank Eng. Mattia Torta for the support provided during the experimental activities.

Author contribution N. Carbone conceived the methodology, implemented it and wrote the paper. L. Bernini designed and performed the experiments, conceived and implemented the methodology, wrote and revised the paper. P. Albertelli conceived the research, designed the experiments, performed them and supervised the research. M. Monno performed the proofread of the paper.

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. This work was developed in the DIGIMAN project, funded by "Asse 1 – Azione 1.2.2 POR-FESR 2014 – 2020 Emilia-Romagna".

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Wang J, Ma Y, Zhang L, Gao RX, Wu D (2018) Deep learning for smart manufacturing: methods and applications. J Manuf Syst 48:144–156. https://doi.org/10.1016/j.jmsy.2018.01.003
2. Mehta P, Rao P, Wu Z, Jovanović V, Wodo O, Kuttolamadom M (2018) Smart manufacturing: state-of-the-Art Review in Context of Conventional and Modern Manufacturing Process Modeling, Monitoring and Control. https://doi.org/10.1115/MSEC2018-6658
3. Kim DH, Kim TJ, Wang X, Kim M, Quan YJ, Oh JW, Min SH, Kim H, Bhandari B, Yang I, Ahn SH (2018) Smart machining process using machine learning: a review and perspective on machining industry. Int J Precision Eng Manuf-Green Technol 5:555–568. https://doi.org/10.1007/s40684-018-0057-y
4. Weimer D, Scholz-Reiter B, Shpitalni M (2016) Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. CIRP Ann 65:417–420. https://doi.org/10.1016/j.cirp.2016.04.072
5. Park JK, Kwon BK, Park JH, Kang DJ (2016) Machine learning-based imaging system for surface defect inspection. Int J Precision Eng Manufact-Green Technol 3:303–310. https://doi.org/10.1007/s40684-016-0039-x
6. Haralick RM, Shanmugam K, Dinstein I (1973) Textural features for Image Classification. IEEE Transactions on Systems Man, and Cybernetics SMC-3, 610–621, https://doi.org/10.1109/TSMC.1973.4309314
7. Lehtomäki M, Jaakkola A, Hyyppä J, Lampinen J, Kaartinen H, Kukko A, Puttonen E, Hyyppä H (2016) Object classification and recognition from mobile laser scanning point clouds in a road environment. IEEE Trans Geosci Remote Sens 54:1226–1239. https://doi.org/10.1109/TGRS.2015.2476502
8. Meng Z, Fan X, Chen X, Chen M, Tong Y (2017) Detecting small signs from large images. In: Proceedings - 2017 IEEE International Conference on Information Reuse and Integration, IRI 2017 2017-January, pp 217–224. https://doi.org/10.1109/IRI.2017.57
9. Gadelmawla ES (2004) A vision system for surface roughness characterization using the gray level co-occurrence matrix. NDT & E International 37:577–588. https://doi.org/10.1016/j.ndteint.2004.03.004
10. Song K, Yan Y (2013) A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. Appl Surf Sci 285:858–864. https://doi.org/10.1016/j.apsusc.2013.09.002
11. Mentouri Z, Moussaoui A, Boudjehem D, Doghmane H (2018) Steel strip surface defect identification based on binarized statistical features. Scientific Bulletin Series B: Chemistry and Materials Science, 80(4)
12. Tao X, Zhang D, Ma W, Liu X, Xu D (2018) Automatic metallic surface defect detection and recognition with convolutional neural networks. Appl Sci, 8. https://doi.org/10.3390/app8091575
13. Fu G, Sun P, Zhu W, Yang J, Cao Y, Yang MY, Cao Y (2019) A deep-learning-based approach for fast and robust steel surface defects classification, vol 121
14. Mentouri Z, Moussaoui A, Boudjehem D, Doghmane H (2020) Steel strip surface defect identification using multiresolution binarized image features. J Fail Anal and Preven 20(6):1917–1927. https://doi.org/10.1007/s11668-020-01012-7
15. Mentouri Z, Doghmane H, Moussaoui A, Bourouba H (2020) Improved cross pattern approach for steel surface defect recognition. Int J Adv Manuf Technol 110(11-12):3091–3100. https://doi.org/10.1007/s00170-020-06050-x
16. Chen S, Chou E, Yang R (2018) The Price is Right. Predicting Prices with Product Images. arXiv: https://doi.org/10.48550/https://doi.org/10.48550/
17. Li B, Zhang H, Ye P, Wang J (2020) Trajectory smoothing method using reinforcement learning for computer numerical control machine tools. Robot Comput Integr Manuf, 61. https://doi.org/10.1016/j.rcim.2019.101847
18. Kassim AA, Mian Z, Mannan MA (2006) Tool condition classification using Hidden Markov model based on fractal analysis of machined surface textures. Mach Vis Appl 17:327–336. https://doi.org/10.1007/s00138-006-0038-y
19. García-Ordás MT, Alegre-Gutiérrez E, Alaiz-Rodríguez R, González-Castro V (2018) Tool wear monitoring using an online,

automatic and low cost system based on local texture. Mech Syst Signal Process 112:98–112. https://doi.org/10.1016/j.ymssp.2018.04.035
20. Hou L, Samaras D, Kurc TM, Gao Y, Davis JE, Saltz JH (2016) Patch-based convolutional neural network for whole slide tissue image classification. 2424–2433. https://doi.org/10.1109/CVPR.2016.266
21. Ding C, Choi J, Tao D, Davis LS (2016) Multi-directional multi-Level dual-Cross patterns for robust face recognition. IEEE Trans Pattern Anal Mach Intell 38(3):518–531. https://doi.org/10.1109/TPAMI.2015.2462338
22. Gopaluni B, Tulsyan A, Chachuat B, Huang B, Lee JM, Amjad F, Damarla SK, Kim JW, Lawrence NP (2020) Modern machine learning tools for monitoring and control of industrial processes: a survey. IFAC-PapersOnLine 53:218–229. https://doi.org/10.1016/j.ifacol.2020.12.126
23. Azimi M, Eslamlou AD, Pekcan G (2020) Data-driven structural health monitoring and damage detection through deep learning: state-of-the-art review. Sensors, 20. https://doi.org/10.3390/s20102778
24. Fix E, Hodges J (1951) Discriminatory analysis: nonparametric discrimination: consistency Properties. USAF School of Aviation Medicine
25. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501. https://doi.org/10.1016/j.neucom.2005.12.126
26. Sharma A, Liu X, Yang X, Shi D (2017) A patch-based convolutional neural network for remote sensing image classification. Neural Netw 95:19–28. https://doi.org/10.1016/j.neunet.2017.07.017
27. Wu J, Ye Y, Chen Y, Weng Z (2018) Spot the difference by object detection. arXiv e-prints
28. Breiman L (2001) Random Forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324
29. Griffiths D, Boehm J (2019) A review on deep learning techniques for 3D sensed data classification. Remote Sens, 11. https://doi.org/10.3390/rs11121499
30. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. 770–778. https://doi.org/10.1109/CVPR.2016.90
31. Huang G, Liu Z, van der Maaten L, Weinberger KQ (2017) Densely Connected Convolutional Networks. 2261–2269. https://doi.org/10.1109/CVPR.2017.243
32. Tan M, Le QV (2019) EfficientNet: rethinking model scaling for convolutional neural networks. ArXiv https://doi.org/10.48550/ARXIV.1905.11946
33. Howard A, Sandler M, Chu G, Chen LC, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V, Le QV, Adam H (2019) Searching for MobileNetV3. arXiv https://doi.org/10.48550/ARXIV.1905.02244
34. Wu S, Zhang M, Chen G, Chen K (2017) A new approach to compute CNNs for extremely large images. Association for Computing Machinery New York, pp 39–48. NY, USA. https://doi.org/10.1145/3132847.3132872
35. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. Institute of Electrical and Electronics Engineers (IEEE), 248–255. https://doi.org/10.1109/CVPR.2009.5206848
36. Krizhevsky A (2009) Learning Multiple Layers of Features from Tiny Images
37. LeCun Y, Cortes C (2010) MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Accessed 20 May 2022
38. Chinchor N (1992) MUC-4 Evaluation metrics. Association for Computational Linguistics, 22–29. https://doi.org/10.3115/1072064.1072067
39. Sasaki Y (2007) The truth of the F-measure
40. Bellman R (1966) Dynamic programming. Science 153:34–37. https://doi.org/10.1126/SCIENCE.153.3731.34
41. Jolliffe I (2011) Principal component analysis, 1094–1096 springer berlin heidelberg. Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_455
42. Hall-Beyer M (2017) GLCM Texture. A Tutorial 3:0. https://doi.11575/PRISM/10182
43. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. Association for Computing Machinery, 144–152. https://doi.org/10.1145/130385.130401
44. Probst P, Boulesteix AL (2018) To tune or not to tune the number of trees in random forest. J Mach Learn Res 18:1–18
45. Smith LN, Topin N (2019) Super-convergence: very fast training of neural networks using large learning rates. https://doi.org/10.1117/12.2520589

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.