A Thesis for the Degree of Ph.D. in Engineering

Cutting Tool Parameter Identification and Machined Surface Quality Inspection Based on Deep Learning for Advanced Machine Tool Development

August 2019

Graduate School of Science and Technology
Keio University

Achmad Pratama Rifai

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols

Index	Unit	Description
Surface quality inspection		
y	µm	height/depth of surface profile
s		number of sampling
L	mm	evaluation length
l	mm	sampling length
Ra	µm	arithmetic average roughness
Rt	µm	total height of the roughness profile
Rz	µm	maximum height of the roughness profile
Rp	µm	maximum profile peak height
Rv	µm	depth of lowest valley
Rq	µm	average roughness square deviation of the profile
Ga		average arithmetic of gray level
F1		major peak frequency
F2		principal component magnitude squared
IACX		integrated autocorrelation coefficient in the *x*-direction
IACY		integrated autocorrelation coefficient in the *y*-direction
IACXY		integrated autocorrelation coefficient in the diagonal direction
HA		average peak spectral intensity coefficient
HI		integrated peak spectral intensity coefficient
σ		standard deviation
Fd		gradient factor of the surface
Ctm		average cycle of texture
C	m/min	cutting speed
S	rpm	spindle speed
F	mm/rev, mm/min	feed rate
D	mm	depth of cut
Image processing and data augmentation		
A		index of fragment (sub-image)
n		number of fragments (sub-images)
σ		amplitude of edges parameter
α		smoothing parameter
β		tone mapping parameter
g		pixel value from the Gaussian pyramid
r		remapping function

Index	Unit	Description
Convolutional neural network		
A_i		obtained accuracy of deep learning model
A_f		accuracy of tool identification using FDDM
A_d		desired/target accuracy
L		loss function value
n		total number of sample (image)
b		batch index
B		batch size
s		iteration index
S		number of iterations
e		epoch
E		number of epoch
w		weight
λ		regularization factor
lr		learning rate
u		class index
C		number of classes
η		momentum hyperparameter
ε		decay step of the learning rate
c		decay factor of the learning rate
p		predicted value
q		target value
δ		Huber hyperparameter
k		filter index
Ch		decision variable for vibration existence
N		output of unstable cutting marks identification problem neuron in continues numerical value
Feature detection, description, and matching		
t		test image
T		number of predictions used to select the candidates
k		class index
im		image index
M		number of image (in each class)
δ		number of obtained inliers or matching points
λ		decision threshold of FDDM
τ		distance threshold of feature matching
β		confidence of feature matching

Index	Unit	Description
Dimension estimation and measurement		
i		index of detected tool
I		number of detected tool in an image
N		neighborhood
T		tool tip
l_a	mm	estimated dimension
l_b	pixels	estimated dimension
s		scale of image
α	°	capture angle
L	mm	length/overhang
d	mm	diameter
h	mm	height
r	mm	sensor displacement
Performance evaluation		
E		error rate
γ_1		Top-1 accuracy of classification task
γ_5		Top-5 accuracy of classification task
A_{Ra}		accuracy of *Ra* prediction
A_{ch}		accuracy of unstable cutting marks identification
A_{Rz}		accuracy of *Rz* prediction
R^2		coefficient of determination
d		degree of the polynomial
t		test image
J		number of test image
n		class index
N		number of classes
T	h	training time
τ	s	test/prediction time

Abbreviations

Abbr.	**Description**
AI	artificial intelligence
ANN	artificial neural network
Avg pool	average pooling layer
BRISK	binary robust invariant scalable keypoints
CAD	computer aided design
CDSM	color distribution statistical matrix
CNN	convolutional neural network
CPU	central processing unit
Conv	convolutional layer
FC	fully connected layer
FDDM	feature detection, description, and matching
GLCM	gray level co-occurrence matrix
GPU	graphical processing unit
GUI	graphical user interface
ILSVRC	ImageNet Large Scale Visual Recognition Competition
IoT	internet of things
JIS	Japan Industrial Standard
LMP	largest matching point
MAE	mean absolute error
MAPE	mean absolute percentage error
Max pool	max poling layer
MP	matching point
MSAC	M-estimator sample consensus
MSE	mean squared error
NN	neural network
NC	numerical control
RANSAC	random sample consensus
ReLU	rectified linear unit
RGB	red green blue (image format)
RMS	root mean squared
RMSE	root mean squared error
ROI	region of interest
R-CNN	regional convolution neural network
SURF	speedup robust feature
SVM	support vector machine

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and most sincere appreciation to Professor Hideki Aoyama, my research advisor and the chairperson of thesis committee, for his continuous support and valuable guidance throughout this challenging and rewarding research work. His comments and advice during the research have contributed immensely toward the success of this study.

I also indebted to my thesis co-supervisors: Professor Yan Jiwang, Professor Yoshimitsu Aoki, and Associate Professor Yasuhiro Kakinuma for their invaluable advices, suggestion, comments and support in this study.

I would also thank the Japan International Cooperation Agency (JICA) for the scholarship and financial support during my three-years study in Japan. My sincere gratitude also goes to the ASEAN University Network/ Southeast Asia Engineering Development Network (AUN/ SEED-Net) for the indirect support throughout these years.

I would like to thank Keio University for providing sufficient environment and facilities to support the study, as well as the staff of manufacturing center for their assistance during this study. My warmest regard goes to every member of Hideki Aoyama laboratory, both graduates and students.

Last but not the least, my gratitude goes to my beloved parents, brother, grandparents, all family members, and friends for their loving encouragement and best wishes throughout the period of my study.

August 2019

Achmad Pratama Rifai

ABSTRACT

The adoption of artificial intelligence (AI) in the manufacturing domain has revolutionized production processes by introducing a higher level of automation, precision, operational efficiency, and productivity in machining. Specifically, in the field of machining inspection, AI has contributed to improving evaluation accuracy and flexibility. In addition, the use of AI has reduced the time spent for decision making by providing the operator with relevant information and facts filtered out from huge repositories of stored knowledge.

The development of AI intersects with machine vision applications. Machine vision has been successfully implemented for monitoring manufacturing systems by considering the properties of each component in the machining process, such as machines, sensors, vision devices, and expertise of the technicians. A typical process of machine vision in industrial inspection consists of image acquisition, image processing, feature extraction, and decision-making. However, it has been shown that the feature extraction step requires deep knowledge of feature engineering which a technician may not possess. In addition, this step consumes longer processing times than other steps and this may hinder its application. To cope with this problem, deep learning using convolutional neural network (CNN), which delivers exceptionally strong performance for vision-based identification, may be considered. CNN automates the feature extraction step by embedding the convolution process into the network. However, the use of CNN in the inspection field must consider the unique features of the machining. Therefore, this study aims to develop a deep learning-based identification and prediction method for machining inspection. This method is implemented in two tasks that are typical in machining inspections: (i) tool parameter identification and (ii) machined surface evaluation. Both these problems have distinct characteristics from the perspective of machine vision. Tool parameter identification is concerned with object classification, while machined surface evaluation is related to the task of texture analysis.

The identification of the setting parameters for a cutting tool is important for ensuring an effective and uninterrupted machining process. Such identification is essential for avoiding collisions and determining the tool overhang in vibration detection analysis. The current practice of manually identifying the tool setting parameters may allow the occurrence of undesirable human errors, such as incorrect recognition, which will result in inaccurate identification. Therefore, another aim of this study is to develop a fast, precise, and reliable automatic on-machine tool parameter identification system. The approach comprises three main steps: (i) identifying the class of tool, (ii) estimating its dimensions, and (iii) measuring the dimensions based on the information from previous steps. In the first step, deep CNN models are developed for an image-based tool recognition problem. In the second step, an automatic dimension estimation algorithm based on the edge-detection method is proposed. The outcome of the first two steps is then used in the third step to automatically generate the numerical control code for the measurement process using a contact-based displacement sensor.

In the surface quality inspection problem, many studies on vision-based machined surface evaluation have been conducted by analyzing feed marks on surfaces, which are captured as images. The previous methods used feature extraction to quantify the surface morphology, in which obtained features are used for building a prediction model. However, the extraction is a complex process, requiring an advanced image filtering and segmentation step, which often takes a long time. This goes against the original aim of the vision-based method which seeks a quick and *in situ* surface evaluation. In this study, we propose the use of CNN to evaluate the roughness and vibration marks from an image of surface texture. This method avoids excessive feature extraction since this step is integrated inside the network during the convolution process. Several loss functions for the prediction model are presented and analyzed based on their suitability and accuracy compared to the actual roughness obtained by a stylus-based profilometer. The performance of the proposed model is assessed on the surface finish obtained from conventional machining operations, such as outside diameter turning, and slot and side milling, with various cutting conditions.

The results from the tool parameter identification have shown that the proposed approach can be used to accurately identify the class of tools with an average prediction accuracy of 98.57%. The average processing time per sample was 2.22 s for tool type identification, 41.01 s for length measurement, and 26.92 s for diameter measurement. In the surface quality inspection, the average error of the best model was verified to be within 10% of the actual measured roughness. Furthermore, the unstable cutting mark identification yielded a high level of accuracy with an average of 96% in the three datasets. The computational time for the prediction was within 0.25–3.00 s per sample per target, depending on the complexity of the network. These results confirm the method's practicability for roughness and unstable cutting evaluation in-between processes. The results from the application of the proposed approach to both the aforementioned problems indicate that the developed deep learning models offer promising alternative methods in the inspection domain.

CHAPTER 1

INTRODUCTION

1.1 Background

The ever increasing demand for improving productivity has led to the advancement of automation of production lines. Moreover, the replacement of humans with automated systems is essential for minimizing or eliminating the human error. However, this replacement becomes more complicated when knowledge is involved. Straightforward automated systems or machines cannot easily replace human expertise when a considerable amount of knowledge and understanding of the problem is required to deliver a good decision and judgment. In such cases, an autonomously controlled *smart factory* with the application of various technologies of ubiquitous computing becomes indispensable [1]. The efforts to pursue smart manufacturing for achieving a higher level of operational efficiency, productivity, and automation mark a new period of industrial revolution, referred to as industry 4.0 [2].

Industry 4.0 is triggered by the adoption of computers and automation equipped with autonomous systems and data exchange in the manufacturing technologies. Lasi et al. [3] mentioned that industry 4.0 is defined by two development directions: (i) application-pull and (ii) technology-push. The triggers for application-pull include the requirements for short product development and innovation cycles, individualization on demand, flexibility on product development, decentralization to reduce organizational hierarchies, and resource efficiency. On the other hand, the technology-push includes approaches for further increasing mechanization and automation, digitization and networking of all manufacturing tools, and miniaturization of resources. As such, the fundamental concepts of industry 4.0 revolve around smart manufacturing which holistically is related to digitization and autonomous systems, cyber-physical systems which refer to the integration of digital and physical resources in the production network, and self-organization which comes along with a devolution of the classical production hierarchy towards a decentralized system.

One important constituent of industry 4.0 is the requirement of embedded intelligence as an essential element of the smart factory, which involves the development and implementation of artificial intelligence (AI) on the production floor. Some examples of AI implementations in the manufacturing field can be found in quality inspection, robot arms, and manufacturing systems design etc. Historically, the application of AI in the industrial field often faced limitation due to the lack of compelling evidence that this method would perform consistently [4]. Other hindrances include the perception that AI requires complex infrastructure and high investment, as well as high dependence on the developer's experience and preferences.

AI covers a wide array of knowledge-based methods. One of its applications can be found in image identification which intersects with the field of computer vision. Computer vision is being increasingly applied in the manufacturing field due to its intelligence and ease of applicability. A machine vision application is considerably simpler, cost-effective, and time-efficient compared to the traditional evaluation method that uses sensors and contact-based profilometers. In addition, the infusion of AI into machine vision allows further analysis and decision-making process. Inspired by these benefits, this study presents the development of vision-based prediction methods by employing AI for the evaluation. The study is implemented in two applications in machining, namely cutting tool parameter identification and machined surface quality inspection. The input data for the predictions are the images of the tool in the tool identification problem and the images of the machined surface in the surface quality inspection problem. In both the problems, deep learning models using a convolutional neural network (CNN) are developed as the main prediction method.

The target of the first problem is to identify the specific types of tools and their dimensions. Therefore, the proposed system for tool parameter identification consists of two phases, namely identification of the tools' type and measurement of their dimensions. The first phase is intended to classify specific types of tools and identify the region of interest (ROI) inside the captured image. This system utilizes CNN, *feature detection, description, and matching* (FDDM), and a regional CNN (R-CNN) for identification and localization. The second phase aims to measure the tool's dimensions. First, estimation of the dimensions using machine vision is performed. In the next step, on-machine measurements are performed using a contact-based displacement sensor to evaluate the cutting tool dimensions. The movement of the tool into the sensor is directed based on the generated numerical control (NC) program. The aim of the system is two-fold: confirming that the correct type of tool is installed and detecting undesirable changes in tool dimensions. The former is intended to prevent the incorrect installation of tools whereas the latter is intended to monitor the tool setup.

The targets for the second problem are prediction of surface roughness and identification of unstable cutting marks based on the pattern of the machined surface. Similar to the previous problems, this study also proposes the development of deep learning models for vision-based surface roughness and unstable cutting evaluation. Two systems are presented, separated and combined. In the separated system, the deep learning models are trained individually for each target, whereas the combined system carries out training and prediction processes for both surface roughness and unstable cutting marks simultaneously. The proposed models are examined on three types of data sets: turned, slot milled, and side milled surfaces.

1.2 Literature Review

1.2.1 Artificial Intelligence for Intelligent Manufacturing

The purpose of an intelligent manufacturing system is to perform the manufacturing activities intelligently as if they were being performed by human operators. However, these intelligent systems are not intended to replace their human counterparts completely. They are designed to support the operators by providing them knowledge and relevant facts filtered out from a huge database of stored knowledge, and consequently, increasing the speed of decision making of the operators [5]. Similarly, the operators' expertise and their prior knowledge, as well as the previously available data of the system, are required to design a capable and reliable intelligent manufacturing system. The process of transferring such knowledge requires artificial intelligence techniques to be embedded in the intelligent manufacturing system.

AI is generally defined as a branch of computer science dealing with intelligent behavior of the machines, as opposed to the natural intelligence possessed by humans and animals. Kaplan and Haenlein [6] defined AI as a system's ability to correctly interpret external data, learn from such data, and use those learnings to achieve specific goals and tasks through flexible adaptation. AI has witnessed a rich spectrum of research activities in wide areas that encompass computer vision, speech recognition, natural language processing, robotics, machine learning, etc. Accordingly, various AI tools have been developed to solve problems in the aforementioned application areas, namely evolutionary algorithms for optimization, Bayesian networks for uncertain reasoning, support vector machine (SVM) for classifiers, and artificial neural network and its latest evolution, deep neural network.

Along with the development of new technologies in manufacturing, in both hardware and systems, the industrial implementations of AI are also growing. The application of AI in the manufacturing sector is widespread due to its domain independent characteristics. Expert systems are widely exploited to create intelligent software capable of solving problems in the same way as human experts do. Evolutionary algorithms and swarm algorithms are popular tools for optimization of complex problems, such as manufacturing process planning, simulation of manufacturing systems, and control systems engineering. Some examples of fuzzy logic and neural network implementations include selection of machine tools [7] and conveyors [8]. Figure 1.1 shows some AI techniques and examples of their application in the manufacturing domain.

One of the applications of AI in manufacturing field is related to machining, where it has been successfully developed and integrated in machining monitoring systems. Abellan-Nebot and Subiron [37] reviewed the AI applications in a monitoring system. Their study categorized the AI applications for intelligent manufacturing based on six key points: (1) sensor system, (2) signal processing techniques, (3) sensor signal feature generation, (4) feature extraction, (5) design of experiments, and (6) AI techniques. Bai et al. [38] used a deep neural network for production quality. Their proposed model consisted of deep belief network for unsupervised feature learning and a regression layer on top of the network. Lee et al. [39] used a CNN for fault classification and diagnosis in semiconductor manufacturing process. Their model tailored the receptive field to multivariate sensor signal slides along the time axis to extract fault features. This approach enabled the association of the output of the first convolutional layer with the structural meaning of the raw data, making it possible to locate the variable and the time information that represent the process faults. As indicated by recent literature [38, 39], the usage of deep neural network and CNN for computer vision in manufacturing problems is growing.

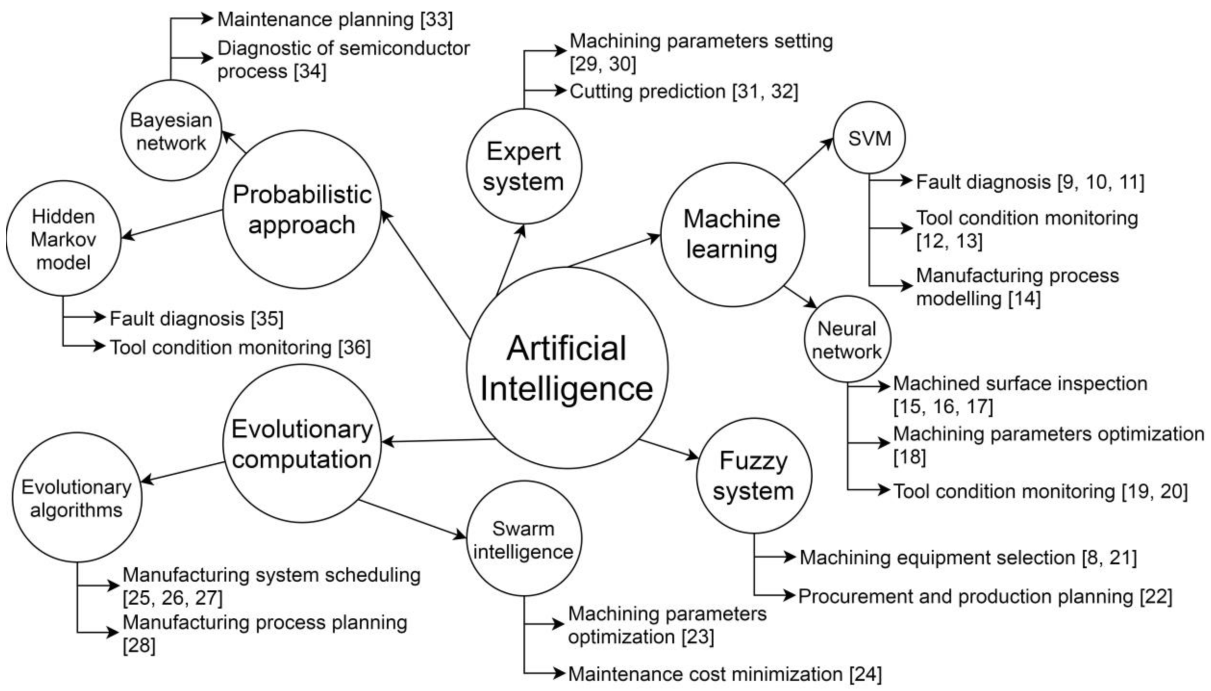

Artificial Intelligence
Probabilistic approach
Bayesian network
Maintenance planning [33]
Diagnostic of semiconductor process [34]
Hidden Markov model
Fault diagnosis [35]
Tool condition monitoring [36]
Expert system
Machining parameters setting [29, 30]
Cutting prediction [31, 32]
Machine learning
SVM
Fault diagnosis [9, 10, 11]
Tool condition monitoring [12, 13]
Manufacturing process modelling [14]
Neural network
Machined surface inspection [15, 16, 17]
Machining parameters optimization [18]
Tool condition monitoring [19, 20]
Fuzzy system
Machining equipment selection [8, 21]
Procurement and production planning [22]
Evolutionary computation
Evolutionary algorithms
Manufacturing system scheduling [25, 26, 27]
Manufacturing process planning [28]
Swarm intelligence
Machining parameters optimization [23]
Maintenance cost minimization [24]

Figure 1.1 AI techniques and their applications in manufacturing

1.2.2 Computer Vision in Industrial Applications

Along with the growing application of visual images, machine vision appears as a viable approach in processing the information of images. It provides a competitive advantage to the industries that employ this technology owing to its ability to improve productivity and quality management [40]. Although overlapping each other, the terms 'computer vision' and 'machine vision' have distinct characteristics. Machine vision traditionally refers to the application of computer vision in industry. Due to its unique association with the manufacturing field, machine vision encompasses a large number of technologies, both software and hardware. Machine vision attempts to build an integrated system while considering the properties of each component in the manufacturing field, such as machines, sensors, vision devices, and expertise of the technicians. Therefore, while computer vision is in the scientific domain, machine vision is concerned more with specific applications of the former to solve engineering problems.

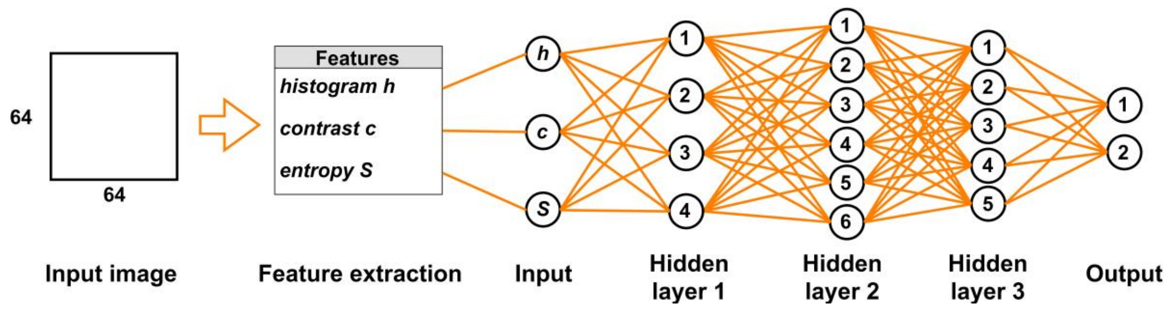
The main activities in computer vision are acquiring, processing, and analyzing the digital image, followed by extracting the features to produce information in categorical, numerical, or symbolic forms. Afterwards, the newly produced information can be used for decision making process. Computer and machine visions have been applied widely in industry. However, it should be noted that there is no universal system that can be applied to vision problems in manufacturing. Instead, the requirements for the design and development of a successful machine vision system vary depending on the application domain, and are related to the tasks to be accomplished, environment, speed, etc. [40]. Therefore, it is important for the designer to fully understand the type of information that computer vision will retrieve from the image, and use in the decision making process.

One of the main uses of computer vision in manufacturing is found in the inspection domain, specifically in dimension calibration [41, 42] and tool condition assessment [43]. Malamas et al. [40] outlined that a typical vision-based industrial inspection process consists of four steps as follows:

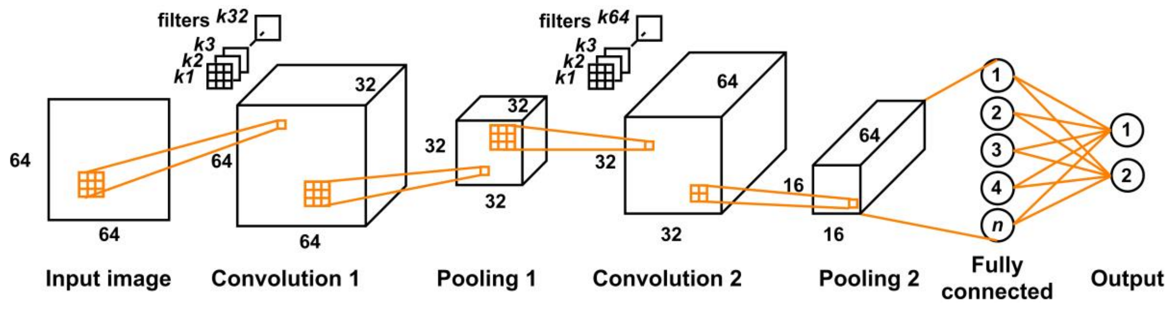
1. Image acquisition: the images that contain the required information are acquired in a digital form.
2. Image processing: the acquired images are filtered and modified to remove the background and unwanted noises. This step also aims to enhance the quality of the image so that the determination of its features becomes easier.
3. Feature extraction: this step extracts a set of known and desired features that describe the image. Such features then can be computed and analyzed by the decision-making method.
4. Decision-making: this final step utilizes the extracted features for reaching a decision by considering the information from the image. To accomplish the same, this step may use statistical and/or AI methods such as a neural network (NN) or a fuzzy system. The decision making process may involve classification or regression problem.

One of the limitations of the traditional computer vision techniques described above is that the feature extraction step requires sufficient knowledge of feature engineering to extract the specific set of required features since it depends on the application domain. Moreover, this step often consumes a long processing time because a significant number of features are usually required to build a decent decision making model. As an alternative, CNN emerged as a method to bypass the feature extraction step. The concept of CNN is explained in the next section, while the detailed model development and its application in both tool parameter identification and machined surface quality inspection problems are discussed in respective chapters devoted to each problem.

Recent studies on inspection and machine monitoring system have hinted that the use of images gives an advantage for both researchers and industry in terms of implementation owing to their ease of usage and simplicity. The image is also noted for its cost-effectiveness since it does not require excessive use of sensors. Due to its advantage, computer vision is an ideal alternative to be applied in both cutting tool parameter identification and machined surface quality inspection problems. The application and the latest development of computer vision in both problems will be discussed in the ensuing sections.



(a) process flow of neural network



(b) process flow of CNN

Figure 1.2 Illustration of the image identification process flows using a regular neural network and CNN

1.2.3 Deep Neural Network

Deep learning is an improvement over artificial neural networks and involves a long chain of networks with multiple layers. One of the deep learning forms is the CNN which is a subset of AI tools that deals with the image as the input data. Thus, the CNN finds applications in computer vision. It shares the properties of its predecessor, namely the neural network, which is made up of neurons that have learnable weights and biases. One main distinguishing aspect of the CNN is that it analyzes the visual imagery directly from the image itself. This is different from the other AI methods where the features of the image are usually extracted first into information that will be fed to the methods. As a result, the CNN uses relatively less image pre-processing and avoids excessive feature engineering. Feature engineering is the process of extracting information (feature) from the raw data, i.e., the image, sound, signal, etc. It is fundamental to the application of machine learning and is both difficult and expensive. Equipped with 3-dimensional neurons in its convolutional layers and shared-weights architecture, the CNN can perform direct image identification without feature engineering. Figure 1.2 shows the difference in the process flow of the CNN and the regular neural network for image identification.

Notice that in the image identification using a regular neural network, the input image must pass through feature extraction process before being fed into the network. In addition to more processing time, this feature extraction process also requires knowledge about feature engineering. Thus, the technicians who are not experienced in feature engineering or computer vision may not be able to modify or update the prediction models. Owing to this issue, the application of regular neural networks for image identification in manufacturing has been very limited. Hence, the CNN has grown to become an important tool of machine vision in manufacturing.

1.2.4 Tool Recognition and Parameter Identification

The requirement of machining complicated and difficult shapes on a single machine stretches the functional scope of the metal cutting machines. As a solution, increase in the number of controlled axes and addition of multi-tasking capability in one chucking process can be attempted, which will enable the machine to manufacture complex products efficiently [44]. These features can be found in multi-tool turning-milling centers. The machines are expected to effectively and consistently maintain their precision, eliminate their own collisions and failure states, and optimize energy consumption [45]. Multi-tool machines are capable of precise production in short periods of time but advances in the computer numerical control (CNC) technology gives rise to an increasing concern over the intelligence of such machines [46]. Integration of more processors and turrets for complex manufacturing increases the complexity and risks of collisions between different tools in multi-tool machines [47]. Furthermore, the tool parameter selection of those machines still heavily relies on the operator’s ability to a great extent. The current technologies are capable of detecting a collision and stopping the machining process only when an undesirable contact happens. Nevertheless, they may be unable to intelligently recognize the tools and provide appropriate feedback. Slight errors, such as inadvertent tool parameter changes, position and dimension changes during the production, and erroneous identification of a wrong tool placed in the turret, may cause collisions and incorrect machining processes. Once collisions happen, they harm the machine and cause a big loss due to the repairing costs and delays in production.

Developing an intelligent tool parameter identification system is critical for reducing the possibility of human error. This has been an active research area in recent years due to the encouragement for automation of the production monitoring systems. Ahmad et al. [47] focused on defining the safe simulated trajectories for virtual manufacturing. The proposed algorithm in their work was developed to generate safe transversal trajectories for virtual machining by detecting and avoiding collisions intelligently. Their method captures the trajectory and the object position in 3D, detects collision, provides a set of solutions to avoid the same, and then selects a feasible solution. Their study was later improved by considering the safe tool path generation in a dynamic environment [41]. The main addition was the ability of the improved method to take into account the evolution of the scene of the obstacle, such as changes in shape, size, and presence during production. Mei and Lee [48] proposed an octree-based collision detection method, to be applied on virtual machine tools and virtual robot arms.

Inconvenient and unreliable manual tool modeling paves the way for the development of automatic tool modeling methods to meet the demand for accuracy, efficiency, and reliability of the system. Zhang et al. [49] worked on this problem by developing a method based on a single view 3D reconstruction algorithm to quickly reconstruct the 3D model of a cutter with tool holder, while they are being installed on the spindle. Their study focused on improving the on-machine camera calibration procedure and developing new algorithms for contour analysis to improve the automation, accuracy, efficiency, and reliability of the tool modeling system. Elgargni et al. [43] extended the tool recognition system by adding a condition monitoring system which is able to identify the tool's current health. This technique used both infrared and visual cameras to track the cutting tool. Collision may also be avoided by automatic checking of machining set up as suggested by Karabagli et al. [50]. Their model utilized chain-processing-based computer vision system to check that the actual machining set-up is in conformity with the desired 3D CAD model that was used to generate the tool trajectory.

Despite a number of studies available in the literature on tool recognition, most studies have not considered the detection of the tool's specific models. This research aims to develop a fast, reliable, and precise automatic identification system for the cutting tool in a turning-milling machine. Tool identification is aimed at eliminating the possibility of human error, and as a consequence, avoiding the collisions. The method consists of two phases: (i) tool type identification and (ii) tool dimension measurement. The first phase is intended to identify the specific types of tools and their relative location in the image. The proposed model combines two approaches, namely CNN and FDDM, both of which have been introduced earlier. The predictions of the CNN are used as the inputs for the FDDM. Subsequently, the prediction method is extended to the R-CNN to localize the relative position of the tool in the image which is essential for the second phase.

The second phase also consists of two steps: (i) dimension estimation using edge detection algorithms and the newly developed approximation algorithm in the first phase, and (ii) on-machine or *in situ* measurement using a contact-based displacement sensor to evaluate the cutting tool dimensions. The identification and measurement phases form a continuous process since the output of the identification process, which constitutes the information of the tool type, is used as an input to automatically generate the NC program for the tool movement during the measurement phase. The aim of the measurement phase is two-fold: confirming that the correct type of tool is installed and detecting undesirable changes in the tool dimensions. The former is intended to avoid an incorrect installation of the tool, whereas the latter is intended to monitor the tool setup.

1.2.5 Surface Quality Inspection Using Machine Vision

Surface quality is crucial in manufacturing industries as it greatly influences the proper functioning of the machined parts. One of the most common indicators of surface quality is surface roughness. Surface roughness is closely related to the fit, wear resistance, fatigue strength, contact stiffness, vibration, and noise of a mechanical part. In other words, it has a significant impact on the service life and reliability of a mechanical product [51]. In the manufacturing industry, surfaces must be within certain limits of roughness to assure specific functional performance. Therefore, accurate measurement is essential to maintain the quality of the machined workpieces. Traditionally, surface roughness measurements have been carried out using a contact-based stylus profilometer. The measurement accuracy of a profilometer is influenced by the radius of its stylus. Although generally it can provide good accuracy, when used to measure surface roughness less than 2.5 µm, a stylus profilometer may introduce a system error owing to the limitation on the magnitude of its radius [16]. In addition, the surface roughness measurement with a stylus involves scratching the specimen surface, which may be undesirable in certain materials. Other drawbacks of this method include long measurement time, additional requirements for certain environments, and a complex setup. These limitations hinder the use of a stylus device for online and quick measurement of the surface roughness.

As an alternative to the stylus-based profilometer, machine vision-based measurement technologies have recently been developed. These methods offer advantages such as high measurement efficiency, non-invasive mode, easy set-up, and no additional requirement for the measurement environment. Research in machine vision-based roughness evaluation mainly revolves around the development of the light source design, image pre-processing techniques, roughness evaluation algorithms and indices, prediction model algorithms, anti-interference ability, and methods for measuring varied processed surfaces [51]. The parameters for estimating the roughness by machine vision can be classified into two-types: non-statistical and statistical indices. The non-statistical indices focus on extracting the geometric features, while the statistical indices include the histogram, the gray level average (*Ga*), and the gray level co-occurrence matrix (GLCM). The non-statistical extraction algorithms for the geometric features can be divided into frequency domain- and spatial domain-based texture feature algorithms. The use of geometric features has been shown to perform well in specimens that have a relatively strong pattern, such as those found in turning operations [52, 53]. However, it has been observed that these indices are unsuitable for specimens with indistinct patterns and a high degree of randomness, as are noticed after a grinding or a milling process. Therefore, statistical features are more commonly used. Among the most popular statistical features is the arithmetic *Ga*, as indicated by previous studies [54, 55, 56]. It is defined as the function of the intensity of the image, in which the mean intensity value is subtracted from individual intensity values according to a set of standards [57]. In predicting the surface roughness, the index *Ga* is seldom used alone. Often, it is combined with other indices, such as machining parameters (cutting speed *C*, feed rate *F*, and depth of cut *D*), GLCM, and second order statistics-based image quantification parameters.

After the extraction, the features are fed to prediction algorithms as input. Based on the type of the decision-making algorithms, the methods can be classified as statistics-based, artificial intelligence, or a combination of both. Statistical analysis with regression has been widely used to correlate the *Ga* of the image and the experimental value of average roughness *Ra* [54]. Kumar et al. [58] added machining parameters (*S*, *F*, and *D*) in addition to *Ga* to build the regression equation. Other statistical techniques commonly used are ANOVA [59], correlation [56,60], and hidden Markov model [61]. Arguably, the most popular technique for texture analysis is GLCM that can derive second order statistical parameters, such as contrast, correlation, energy, homogeneity, and maximum probability. Gadelmawla [62] extracted 24 features based on GLCM and correlated each feature with the average roughness *Ra*. Min et al. [63] studied linear and non-linear regression models using 14 GLCM based extracted features. The variation, as well as the relation, between each texture parameter and *Ra* was analyzed using multiple regression analysis.

In the application of artificial intelligence, SVMs and NNs have been developed to build prediction models. Dutta et al. [64] utilized an SVM with regression for tool condition monitoring by analyzing the turned surface. The model was built using the GLCM-based parameters, such as contrast, dissimilarity, and second diagonal moment, as predictors. Bhat et al. [65] also used an SVM for analyzing the machined surface image using the GLCM features as input. However, their model was developed for a tool wear classification problem. Some researchers have worked in the area of machine vision for classification of both texture and roughness estimation using artificial neural networks (ANNs). Morala-Argüello et al. [66] classified roughness with a multilayer perceptron ANN. The input features were extracted using wavelet transform in the frequency domain. Palani and Natarajan [16] developed an ANN based on 2D Fourier transformed image features for the prediction of roughness in CNC end milling. On the other hand, Dhanasekar and Ramamoorthy [55] used the spatial frequency, *Ga*, and standard deviation as inputs to an ANN.

In addition to the surface roughness estimation, machine vision has also been applied to vibration identification. Khalifa et al. [67] used the *Ga* index applied to edge-enhanced images. Subsequently, texture analysis using the GLCM-based energy and entropy descriptors was carried out to distinguish between the samples showing unstable cutting marks, and those that were produced from stable cutting. Szydłowski and Powałka [68] developed a chatter detection algorithm based on local gradient estimation, while considering machining parameters. However, this approach employed threshold-based decision-making in which the threshold needed to be calibrated for each process. Szydłowski et al. [69] identified unstable cutting marks by observing the speckled pattern over an illuminated region of a milled surface. The light source configuration was the key element in this approach. Similar to their previous research, the decision criteria were threshold-based, which needed to be adjusted for each case. In a recent study, Lei and Soshi [70] attempted to determine the chatter frequency using digital image processing and texture analysis. The frequency of the vibration left on a surface was calculated from the cutting speed and the wavelength of the spatial frequency obtained from the intensity profile.

Table 1.1 Previous literature on vision-based surface roughness prediction

Authors	**Machining**	**Feature extraction method**	**Input data (features)**	**Target (roughness parameter)**	**Prediction method**
Palani and Natarajan [16]	milling	2D Fourier transform	*F1*, *F2*, *S*, *F*, *D*, *Ga*	*Ra*	ANN
Samtaş [17]	milling	-	Black and white (binary) image pixel	*Ra*	neural network
Liu et al. [51]	grinding	CDSM	overlap degree (*S*)	*Ra*	regression
Sarma et al. [54]	turning	2D Fourier transform	*S*, *F*, *D*, *Ga*	*Ra*	linear regression
Kamguem et al. [56]	turning	grey level distribution	*Fd*, *Ctm*, *Ga*	*Ra, Rt, Rz, Rp, Rq*	correlation
Nammi and Ramamoorthy [57]	milling	Fast Fourier transform, GLCM	GLCM parameters (contrast, correlation, energy, maximum probability), *Ga*, fractal dimension	*Ra*	linear regression
Chiou et al. [60]	milling	Grey level distribution	mean, RMS, and σ of intensity level	*Ra*	correlation
Jeyapoovan and Murugan [71]	milling	Metrics Euclidean distance and Hamming distance calculation	the Euclidean and Hamming distances	*Ra*	classification based on the distances
Tootooni et al. [72]	turning	graph theory	Fiedler number $\lambda 2$	*Ra*	regression
Dhanasekar et al. [73]	milling, grinding	monochromatic speckle autocorrelation	*IACX, IACY, IACXY, HA, HI*	*Ra*	correlation, regression
Lu et al. [74]	grinding, milling	CDSM	*Ga*, average power spectrum, CDSM indices	*Ra*	correlation
Suhail et al. [75]	milling	speckle images	mean and σ of line speckle images	*Ra, Rda*	correlation,
Pour et al. [76]	turning, grinding, milling	wavelet transform	entropy, Lyapunov exponent	*Ra*	time series analysis
This study	**turning, milling**	**-**	**image pixel**	***Ra***	**CNN**

Despite the advances in machine vision in the area of roughness identification, most state-of-the-art development in prediction models for either surface roughness or chatter vibration suffer from the same drawback in that good feature extraction methods are needed to generate good input data for the prediction models. The latest research by Samtaş [17] attempted to skip the feature extraction step by converting the image to binary values, and then feeding them directly to an ANN. Nevertheless, this process was computationally expensive; thus it was limited to a narrow network only. Therefore, to fill the gap, this study proposes the development of CNN models as prediction algorithms. Table 1.1 summarizes the previous literature on vision-based surface roughness prediction and the proposed techniques in this study.

Unlike the previous research, the current approach omits the feature extraction process as the input is the image itself. In the traditional methods, feature extraction is performed to find either non-statistical or statistical indices to represent the value of the image. Afterwards, these detected features are used as input to the prediction model. Very often, these feature extraction processes consume long processing time, especially if multiple features are used in the prediction model. Moreover, feature extraction requires complex image processing and a deep understanding of feature engineering to extract good features from the image. Bypassing the feature extraction step not only reduces the processing time, but also simplifies the whole prediction system. However, extra measures should be taken because the CNN process is considerably more complex than machine learning. Often, this results in CNN requiring longer training time due to its deep layers. To obtain and analyze efficient models, in this study, various architectures are explored.

This study proposes the development of deep learning models for vision-based surface roughness predictions and unstable cutting mark evaluation based on the vibration marks. To the best of our knowledge, no research has been performed to explore or develop deep learning models as an alternative for either roughness estimation or identification of unstable cutting marks. Unlike the surface roughness estimation, the unstable cutting mark identification using machine vision is not yet widely developed. Therefore, first two models are developed in this study for these two problems. Image processing methods such as Laplacian filtering and histogram equalization are used to not only enhance the contrast between the ridge and valley regions in the image of the machined surface, but also for data augmentation. A set of experiments in turning and milling machines were conducted to test the accuracy of the system. The machining experiments were performed with various cutting conditions by varying the variables — cutting speed *C*, spindle speed *S*, feed rate *F*, and depth of cut *D*. Afterwards, a combined model, that caters to both the problems, is presented.

1.3 Research Gap

In tool identification problem, most of the previous studies did not consider the detection of the tool's specific model or type. The identification of the type is as important as reconstruction [49] and virtual trajectory generation [47] for avoiding a possible collision.

Moreover, it also serves as double-checking method to confirm that the correct tool is installed and prevent misidentification. The proposed vision-based method ensures that the identification process is uncomplicated without heavy resource requirement.

In the surface quality inspection problem, most literature on vision-based methods followed the four traditional steps: image acquisition, image processing, feature extraction, and decision making. While this scheme has been successful in accurately estimating the surface roughness, it still has some limitations. The feature extraction step often consumes long processing time and requires heavy image processing to get good quality features. Moreover, this step requires knowledge of feature engineering to select appropriate features to be used for decision making. Owing to these issues, the vision-based method is not easy to be re-applied to other data. Since model updating is always required to cover new data, for example, to add surfaces using other materials, the operators must modify the estimation model. With the proposed CNN-based system, the feature extraction step is omitted since the feature is extracted automatically by CNN during the convolutional process. Therefore, the proposed CNN-based system is easier to be re-trained and implemented in production floor since prerequisite knowledge of feature engineering for the operators is unnecessary.

To summarize, this study attempts to implement computer vision and artificial intelligence further in machining domain. The infusion of AI into manufacturing is essential for building an autonomous system that in turn, constitutes one of the concepts of a *smart factory*. By realizing “vision” and CNN, a fully automatic identification system can be achieved. This automatic system can also be extended to tool dimension estimation and measurement. Therefore, the proposed system offers an alternative with less interference of the operators in its operation compared to its predecessors.

1.4 Research Objectives

This research aims to develop intelligent and automatic vision-based inspection systems for decision making with minimum human intervention during the prediction process. The proposed system utilizes CNN as the core prediction method to solve the two aforementioned problems in manufacturing, namely (i) tool recognition and parameter identification, and (ii) surface quality inspection. Accordingly, a number of objectives have been set to accomplish the solutions for each problem. First, the objectives for tool recognition and parameter identification problem are set as follows:

- Developing a fast, precise, and robust automatic recognition system for tool type identification.
- Developing a vision-based tool dimension and overhang estimation system.
- Constructing a procedure for *in situ* tool dimension measurement using contact-based displacement sensor.

Subsequently, the objectives for the surface quality inspection problem are set as follows:

- Developing a vision-based method of machined surface roughness estimation.
- Developing classification and decision-making models for the identification of unstable cutting marks based on the machined surface quality.
- Formulating combined models that can simultaneously perform estimation of the surface roughness and identification of unstable cutting marks from the machined surface.

1.5 Organization of the Dissertation

This chapter has laid the background of the study and the importance of developing intelligent systems in the manufacturing field in accordance with the vision of 'Industry 4.0'. Subsequently, literature review has been presented to give an insight about the current developments in AI implementation for industrial applications, especially tool parameter identification and surface quality inspection, which are the focus areas in this study. Moreover, the state-of-the-art development of computer vision in manufacturing has been described. This chapter has also explained the gaps in the previous research and introduced the objectives of this study.

Chapter 2 discusses the cutting tool parameter identification problem and the proposed system to solve this problem. This chapter elaborates the detailed flow of the proposed system that begins with the tool type identification based on imaging and ends with the tool dimension measurement. In addition, various architectures and configurations of CNN that constitute the core of the prediction models are described.

Chapter 3 covers the second problem, namely the machined surface quality inspection. It discusses the two goals essential for solving this problem, which are estimation of surface roughness and evaluation of the unstable cutting. Accordingly, the proposed image processing and deep learning models for this problem are explained. Furthermore, the proposed combined model that can address simultaneously both the goals with a view to increasing the model efficiency is introduced.

Chapter 4 presents the experiments on both cutting tool parameter identification and machined surface quality inspection problems. The chapter discusses various aspects of the experiments, i.e., the data used, the performance evaluation measures, the experimental setting, and the results. Furthermore, analysis of results is presented, on both the effectiveness (accuracy) and the efficiency (processing time) of the proposed models.

Chapter 5 presents the conclusion of the study by summarizing the advantages and limitations of the proposed system. Finally, this chapter explores avenues for future work by considering the potential improvements in data acquisition, method development, implementation, and expansion of the objectives.

CHAPTER 2

CUTTING TOOL PARAMETER IDENTIFICATION

2.1 Tool Parameter Identification Problem

The automatic identification of a cutting tool is important, especially in modern computer numerical control (CNC) machining centers, which are equipped with a large number of tools. Such an environment can be found, for example, in a turning-milling machine with multiple tools of several types. Errors, such as incorrect disposition, mistakes in recognizing tools, and lack of proper communication feedback in such machines act as threats to the machining process and potentially cause collisions and production delays.

The current day computer-aided manufacturing (CAM) software is capable of preventing the collisions, while the latest machines are designed to halt the machining process once an undesirable contact occurs. However, prior information of the tool parameters and the dimensions are still required as inputs in such a system. Therefore, to obtain this information, this research is aimed at developing a fully automatic identification system for cutting tools in turning-milling machines so that the possibility of human error may be minimized, and the occurrence of collisions may be prevented. Besides preventing collisions, the tool identification and measurement mechanism is also intended to measure the tool overhang, which influences the vibration occurring during the machining.

2.2 Framework of Tool Parameter Identification System

This chapter introduces the approach to classify, predict the type, and then measure the dimension of the cutting tools. The overall system consists of two phases, namely type identification and dimension measurement. In the first phase, considering the ease in obtaining the data, the type identification method uses images of the tools as the input. Another rationale behind using the images is that the image-based recognition is considerably faster than the other approaches, such as point cloud and 3D scanning methods. In addition, an image does not require any sophisticated device; a digital camera is adequate to capture the information, thus reducing the investment cost for such a system.

The tool type identification phase begins with the type prediction using a CNN. For optimum accuracy, the recognition model is then extended by adding the FDDM process. This process uses the predictions of the CNN as its input. The FDDM step is performed to validate the prediction of the CNN and to improve the overall accuracy of the method. The framework for the development of the recognition model combines both the methods. Afterwards, an R-CNN models are developed for ROI identification. Figure 2.1 shows the flow diagram of the development of the proposed tool type identification method.

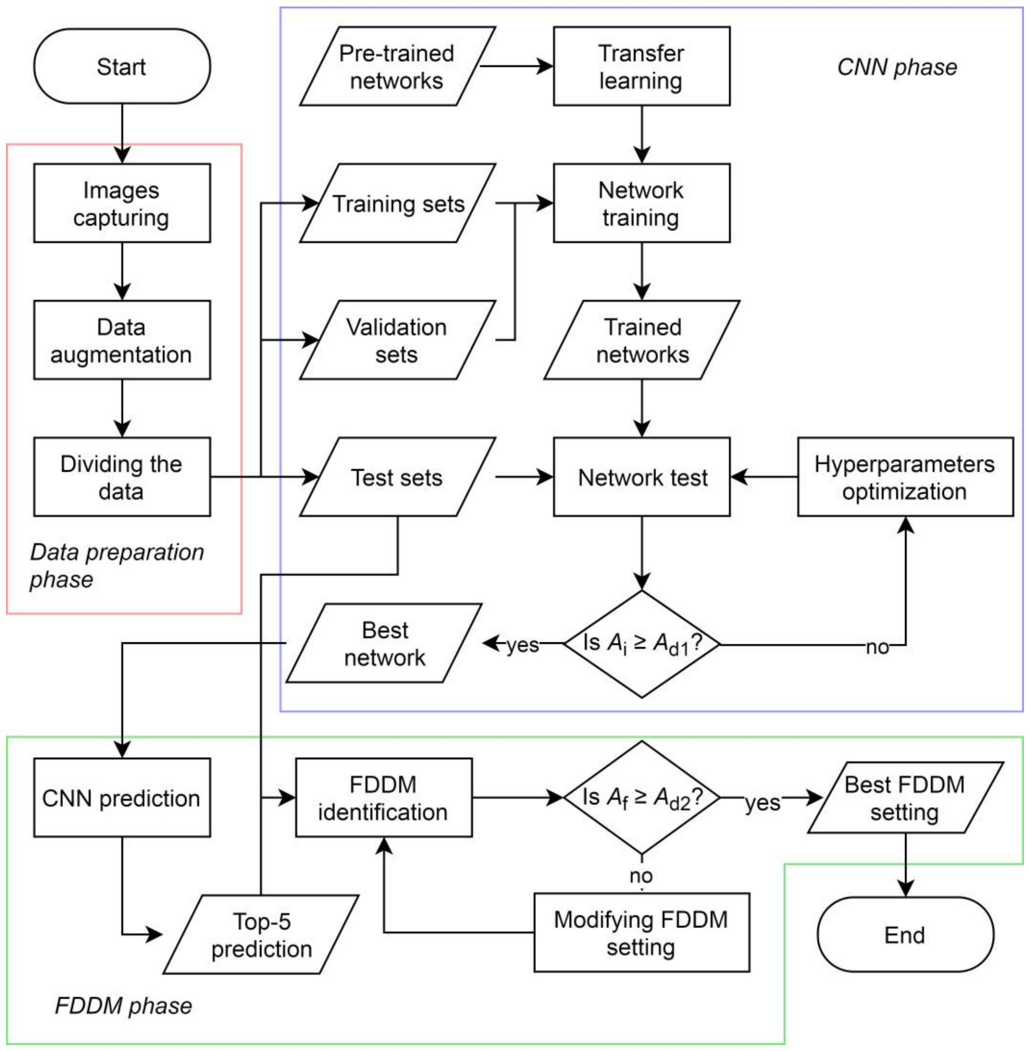


Figure 2.1 Framework for the development of the tool type identification method

The development of the method starts with the data preparation which includes image capturing, data augmentation, and dividing the data into three sets (training, validation, and test datasets). Transfer learning of the CNN is performed using several pre-trained networks. The training and validation sets are fed to the newly modified networks in the training process to build the trained networks. Meanwhile, for the full learning approach, the development starts from crafting the architecture and determining the optimum configurations. The networks are then judged on whether they have achieved satisfactory performance by comparing their accuracy A_i to the desired accuracy A_{d1}. The accuracy is exclusively calculated from the network testing process using the test set. If the accuracy A_i

is less than desired accuracy A_{d1}, the optimization of hyperparameters is carried out and the training process is repeated using new hyperparameter values. Once $A_i \geq A_{d1}$, the network is considered to be adequate for the next process.

The test set is once again used in the FDDM identification process, which also uses the "Top-5" prediction outputs of the CNN identification. The second decision making is then undertaken to analyze the performance of the method. If the obtained accuracy of the FDDM process A_{f1} is less than the desired accuracy of this phase A_{d2}, the parameters are modified, and the identification process is repeated until $A_{fi} \geq A_{d2}$. Once the model development phase is complete, its outputs are ready to be used in the real identification process, which is the end user program. The complete identification process is described in Fig. 2.2.

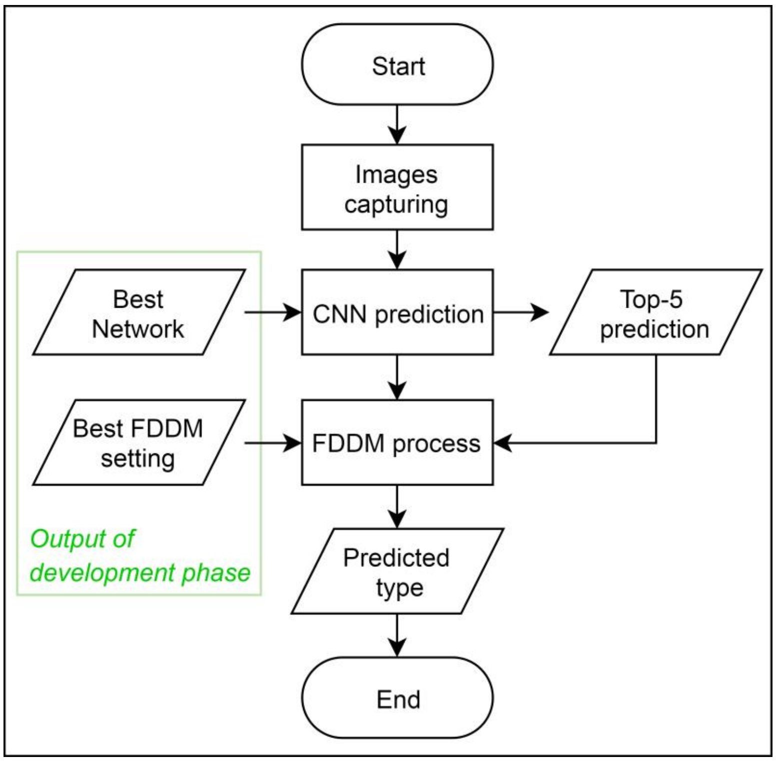


Figure 2.2 Flowchart of the tool type identification process

The input of the end user identification process is comprised of the new images of the tools. The two main processes of this phase, the CNN prediction and the FDDM identification, are identical to the development phase. The CNN prediction and the FDDM identification utilize the two outputs of the model development phase, namely the best network and the best FDDM setting, respectively. Afterwards, the object localization using the R-CNN is performed to obtain the ROI. Eventually, the final output of the recognition method is the predicted type of the tool and its localized ROI.

The dimension measurement phase involves two main processes, namely estimation and measurement. Computer vision is used for estimating the tool dimensions, which, in turn, will be used for generating an automatic NC program. The input of this process is the image inside of the ROI that has been identified by the R-CNN. The generated NC program is then used for the tool movement during the measurement process using a contact-based displacement sensor. The flow of the tool parameter identification system is depicted in Fig. 2.3. The whole process of tool parameter identification, from image capturing to dimension data acquisition, is performed inside the machine. Hence, the proposed system aims for an *in situ* inspection, obviating the need for the tool removal from the machine and the holder.

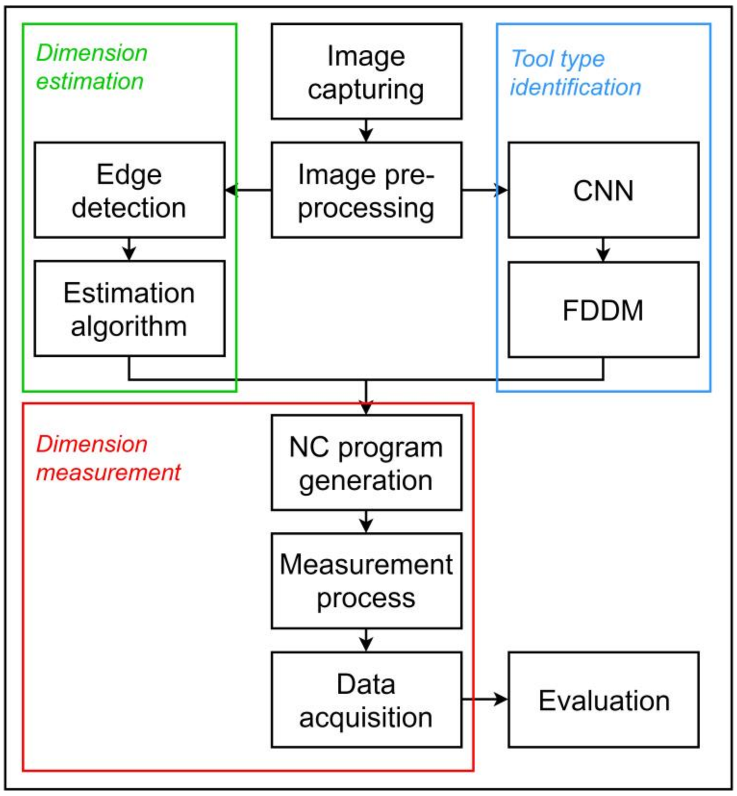


Figure 2.3 Flowchart of the overall tool parameter identification system

2.3 Convolutional Neural Network (CNN) for Tool Identification

The first step of the system is the identification of the tool class (type). The class represents the specific type of the tool. For example, two end milling tools with different number of flutes or dimensions are classified into two different classes. The vision-based identification process is performed by using digital images for their ease of use, low cost, and fast processing. CNN is selected as a classification method owing to its robust accuracy in image classification and recognition. Hence, this study adopted CNN for the machining tool type identification.

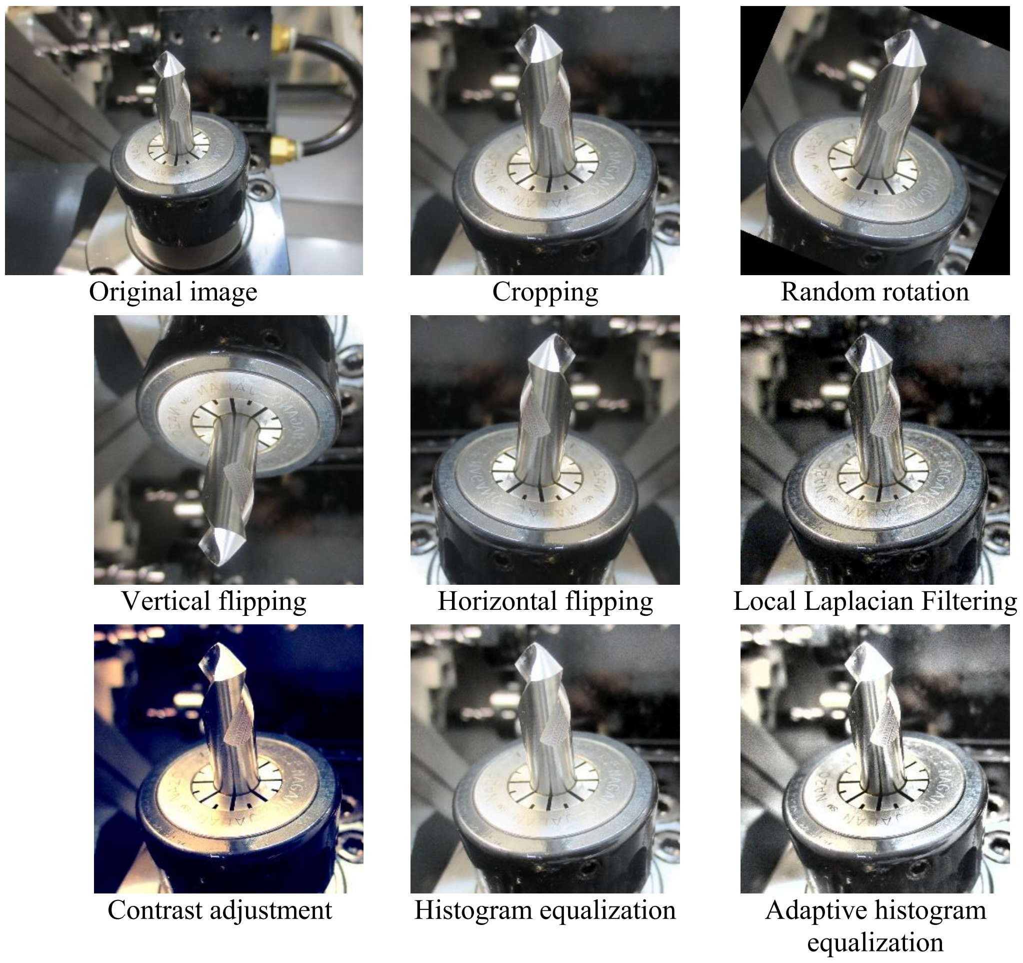


Figure 2.4 Image pre-processing techniques used for data augmentation

2.3.1 Data Augmentation Process

Owing to the requirement of a large number of images for deep learning training, data augmentation is used to artificially multiply the obtained images. Previous research studying the benefits and limitations of data augmentation demonstrated that this method could improve performance in an imbalanced class problem, and regularize overfitting [77]. Eitel et al. [78] noted that the adapted network trained with data augmentation outperformed the baseline model, clearly indicating that additional domain adaptation is necessary for robust recognition in real-world scenes.

There are various techniques that can be used for data augmentation, for example, orientation modification techniques and image enhancement techniques. Orientation modification techniques, such as random rotation, and horizontal and vertical flipping, are widely used as standard methods. Other alternatives include image enhancement techniques, such as contrast adjustment, local Laplacian filtering, histogram equalization, and contrast-limited adaptive histogram equalization. Figure 2.4 displays examples of the image pre-processing methods for data augmentation in this study. Beforehand, the image is center-cropped around the tool and tool holder position since it is the target region for later processes. Furthermore, it minimizes the undesired effect of the background during the classification and the prediction process.

2.3.2 Architecture of the CNN

The advent of CNN has brought in a radical improvement in the field of image identification and classification. CNNs are trainable multistage architectures generally composed of a combination of a convolutional layer, non-linearity layer, and feature pooling layer [79]. The final stage of a CNN consists of a series of fully connected layers and an output layer which are similar to a regular neural network. The receptive field of a convolutional unit with a given weight vector (filter) is shifted step-by-step across a 2D array of input values, and in our case, the pixels of an image. The resulting 2D array of subsequent activation events of this unit can, then, provide inputs to higher-level units. The CNN is trained using a back-propagation algorithm by minimizing the cost function with respect to the weight *w*, as described in Eq. (2.1).

$$L = -\frac{1}{n}\sum_{i}^{n} \vartheta_i \log\left(p(y_{iu})\right) \tag{2.1}$$

where L is the cost (loss) function of batch b in iteration s and n is the number of training images. $p(y_{iu})$ denotes the probability of the ith training image *im* to be correctly classified into class u. ϑ_i is a binary variable which has value 1 if and only if the sample *im* belongs to class u. The number of iterations S is the number of images n divided by the batch size B, i.e., $S = \lceil n/B \rceil$. The above function is a cross entropy loss function, commonly used for classification problems. Subsequently, the weight of the next iteration is updated as follows:

$$\begin{aligned} v_{t(s+1)} &= \eta v_{ts} - lr\frac{df}{dw_s} \\ w_{t(s+1)} &= w_{ts} + v_{t(s+1)} \end{aligned} \tag{2.2}$$

where η is the momentum indicating the contribution of the previous weight in the current iteration and w_{ts} is the weight in tth convolutional layer at the sth iteration. The learning rate lr is a hyperparameter that can be updated in each epoch according to the decay learning schedule.

The developed models use stochastic gradient descent with momentum as the optimization algorithm. The addition of momentum improves the speed of convergence by accelerating the gradient vectors in the right direction. Limited available data in this study may cause the models to be susceptible to overfitting. Here, regularization is used to reduce the possibility of overfitting. Regularization works by adding extra measure into the cost function. This extra term is a function of the weights *w,* and the regularization factor λ, which governs how much influence the weights have in the cost function. Equation (2.3) formulates the new loss functions with regularization.

$$L = -\frac{1}{n}\sum_{i}^{n}\vartheta_i \log\left(p(y_{iu})\right) + \frac{\lambda}{2n}\sum_{w} w^2 \tag{2.3}$$

The distinct advantage of the deep learning method over the traditional machine learning is that it directly employs an image as the input and does not require a prior feature extraction process, thus reducing the human intervention and shortening the overall process. Figure 2.5 illustrates the process flow of deep learning for the tool identification problem.

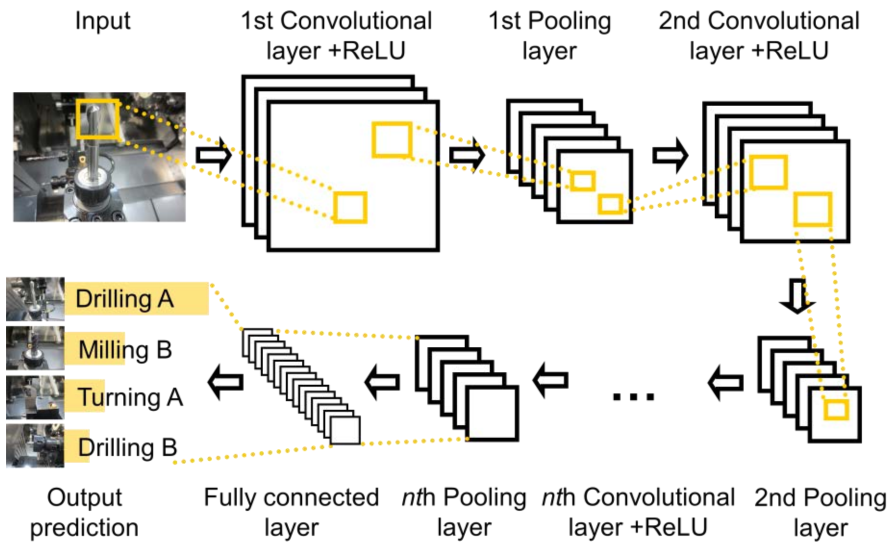


Figure 2.5 Illustration of CNN for tool identification

In a traditional method, feature extraction method is used to identify the feature values of an image, namely the histogram, entropy, contrast, *Ga*, Euclidian distance, etc. On the other hand, in the CNN, the features are extracted automatically by convolutional layers. The goal of a convolutional layer is to filter the pattern. The convolutional layers contain a number of filters with a specific size (i.e., 3×3). As the filter is sliding (convolution process), the filter is multiplied with the pixel value of the input image (dot product, element wise multiplication). Figure 2.6 shows an example of the convolution process, with a filter size of

3×3 and input image size of 5×5. The network has zero padding and one sliding, thus the output will have 3×3 size.

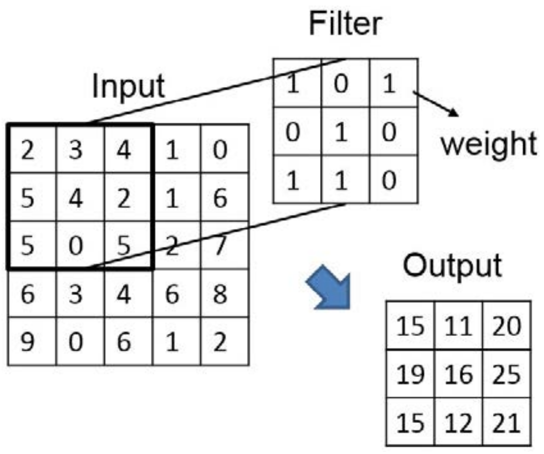


Figure 2.6 Example of the convolution process in a convolutional layer

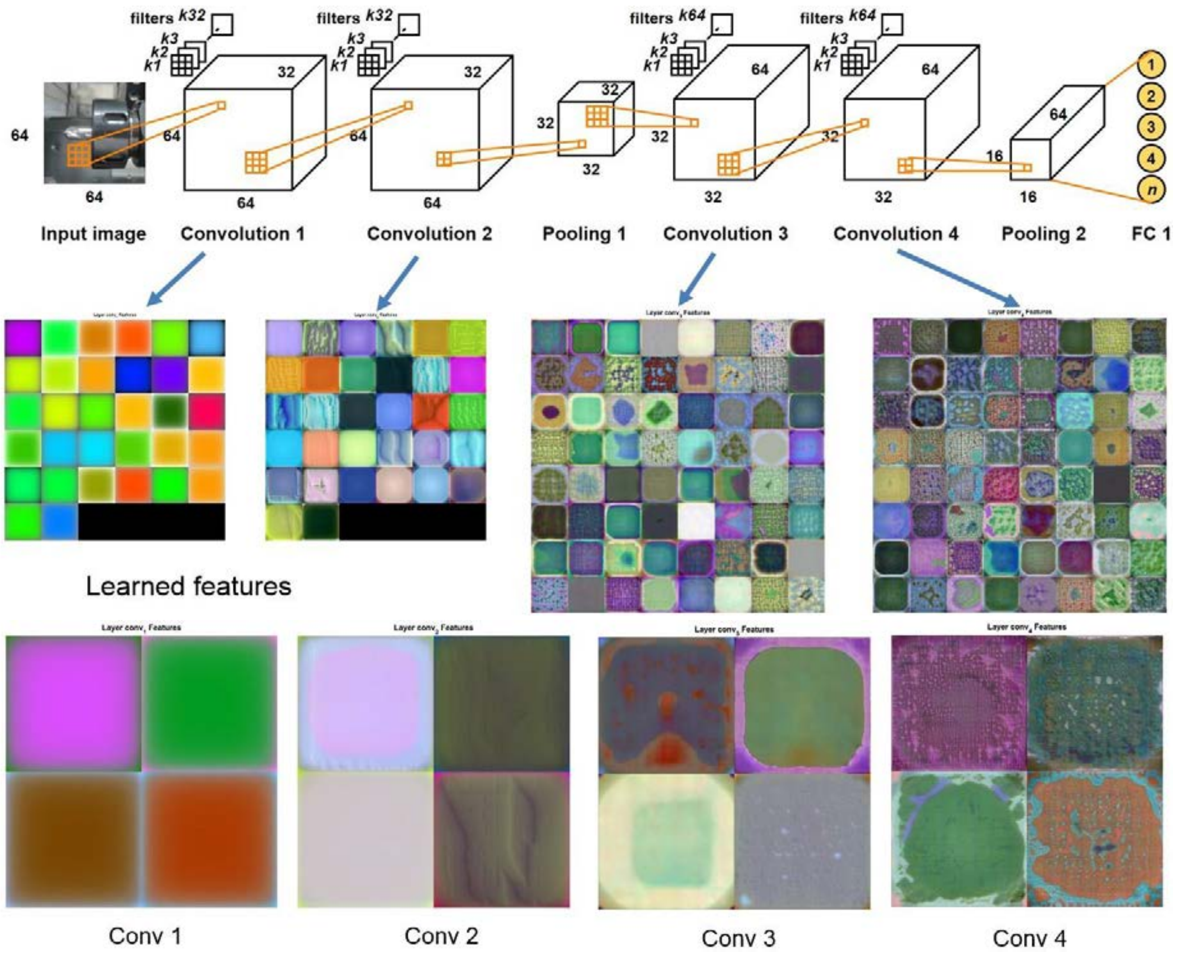
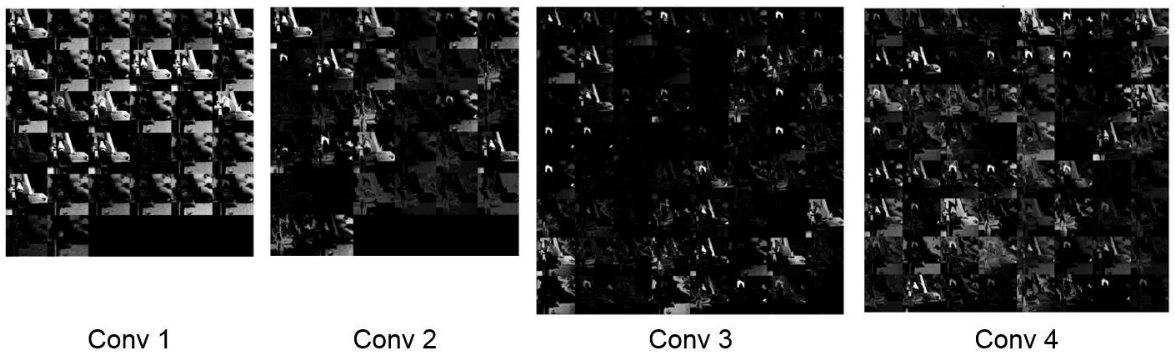
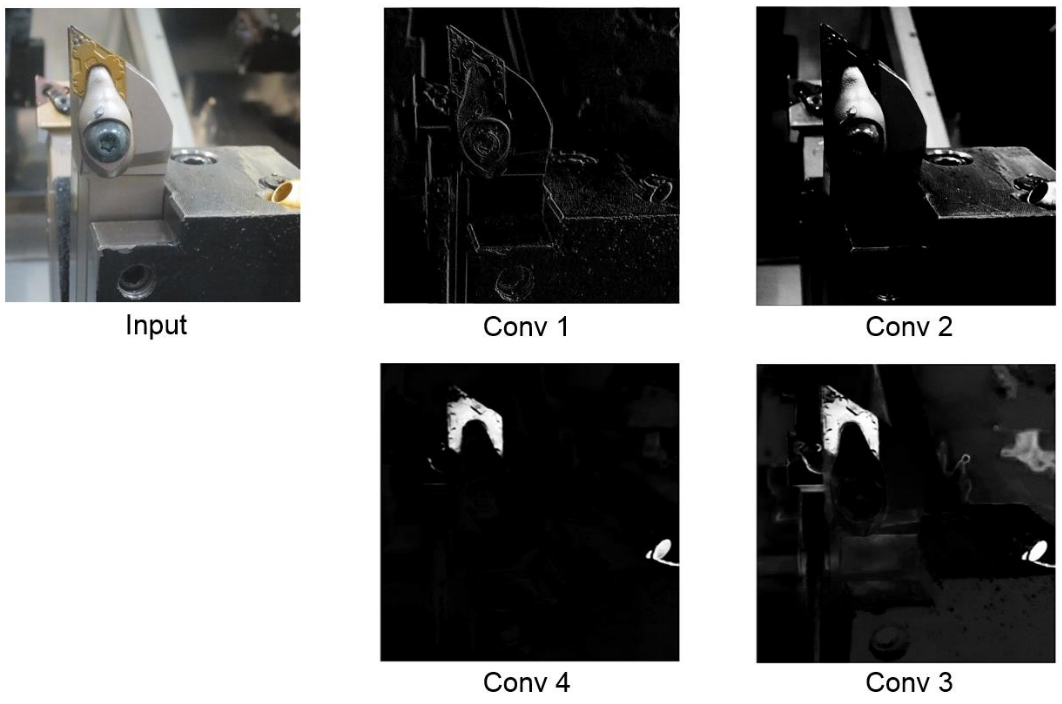


Figure 2.7 Visualization of the learned features for cutting tool identification

Each filter works as a feature identifier, filtering out where the feature exists in the input image. The weight values of the filters are adjusted during the training time using the backpropagation method with stochastic gradient descent according to the value of the loss function. When a weight recognizes a pattern that it has seen before, it returns high values. Each position results in an activation of the neuron. The combination of high weights from various filters lets the network predict the content of an image. The output is collected in the feature map. Figure 2.7 shows visualization of features that have been learned by the network for cutting tool identification problem. The feature maps were visualized by using Deepdream algorithm [80].



(a) activation maps in each convolutional layer



(b) examples of activation in each convolutional layer

Figure 2.8 Visualization of activation in each layer

The network consists of 4 convolutional layers (named Conv 1, Conv 2, Conv 3, and Conv 4) with the number of filters in those layers being 32, 32, 64, and 64 respectively. The number of channels also corresponds to the number of filters. The lower layers (Conv 1 and Conv 2) usually learn simple features such as color and edge, while the higher layers (Conv 3 and Conv 4) learn more complex features, such as shape. Because Figure 2.7 shows the combination features that have been learned by the network using each filter, it looks abstract and is difficult to understand. For a clearer understanding of the feature extraction in the convolutional layers, visualization of activation in each layer is presented in Fig. 2.8.

The number of activation maps follows the number of filters. For example, conv 1 layer has 32 filters and will result in 32 activation maps. The bright pixels show strong positive activation, indicating features that have been learned. As lower layers mainly identify simple features, the activation of Conv 1 layer shows the detected edge feature, while activation of Conv 2 layer highlights the light intensity. Higher layers, i.e., Conv 3 and Conv 4 identify the shape feature, i.e., the shape of the turning tool insert.

2.3.3 Transfer Learning Model Development

As mentioned earlier, this study features the development of CNN models for a tool recognition problem. Two approaches are explored to develop the models: transfer learning and full training from scratch. Full training of the CNN by constructing the network from scratch is often difficult because it requires a vast amount of data. In addition, it needs a great deal of expertise to ensure proper convergence [81]. This obviously causes difficulties in the implementation of the CNN in specific applications with limited data sets. Moreover, training a deep CNN is a time-consuming process, as it requires a repetitive adjustment in the architecture and the learning parameters to avoid overfitting and convergence issues. As an alternative, transfer learning is a promising technique which enables the transfer of weights from a pre-trained network to new networks. Thus, transfer learning is advantageous when the amount of data is limited. After the transfer of weights, the last fully connected layer is replaced with a new layer that conforms to the number of classes in the new data sets and then the new networks are retrained. The transfer of weights from a pre-trained network to the new network conserves good initial weights and biases during the backpropagation process.

Transfer learning is mainly performed by employing previous architectures, especially the ones that have been developed for ImageNet Large Scale Visual Recognition Competition (ILSVRC). ILSVRC is considered as one of the most important competitions in the development of CNN. One of the widely used architectures for transfer learning is Alex-Net by Krizhevsky et al. [82], which was developed in 2012. This architecture consists of five convolutional layers with various sizes of filters, and three fully connected layers. The availability of high specification GPUs has allowed the construction of deeper architectures. Simonyan and Zisserman [83] pushed the prior-art configurations by creating very deep architectures with a depth of up to 19 weight layers, known as VGG-Net. The distinct attribute of this architecture was that it employed uniform and small 3×3 convolution filters in all the layers in contrast to previous architectures wherein various filter sizes were used. In the current study, a pre-trained model of AlexNet is presented and three models of VGG-Net with 11, 16, and 19 weight layers are used for transfer learning. These models are selected owing to their high accuracy, applicability to unfamiliar problems, and computing time.

The MatConvNet deep learning platform, an open source library of CNN for computer vision applications in a MATLAB environment, and VLFeat library, are used for transfer learning and re-training. This library was developed by Vedaldi and Lenc [84]. Algorithm 2.1 expresses the transfer learning process of CNN using pre-trained networks.

Algorithm 2.1 Transfer learning process of CNN

```
1:  Load pre-trained network
2:  Delete the last FC layer
3:  Create a new FC layer according to the number of class of the new
    problem
4:  Load the data
5:  while $e \leq E$
6:     while obtained accuracy $A_i$ < desired accuracy $A_d$
7:        Train the network
8:     end while
9:  end while
```

2.3.4 Full Learning Model Development

Although transfer learning can generally achieve high accuracy with only limited training data, such knowledge transfer may be unsuitable for our target applications owing to the substantial differences between the images of the tool and those used to develop such pre-trained models. Anticipating the occurrence of such a problem, full training from scratch is also considered. Because full training requires a large amount of data, each image is multiplied by a rotation of certain degrees. To analyze the effect of the depth (number of layers) and the design of the layers on the accuracy and efficiency of the models, various CNN architecture configurations are developed as described in Table 2.1.

The architectures consist of several stages. In stages 1–3, each phase comprises two pairs of convolutional layers and rectified linear unit (ReLU) layers. Filter sizes of 3×3 and 4×4 are used. Although the use of even-sized filters is uncommon owing to their asymmetrical properties that alter the output size, it is rationalized that an even-sized filter may provide higher efficiency than its commonly used odd-sized counterpart. The matrix is thus downsized using a 2×2 pooling layer. In the last stage, a series of fully connected layers is used. To avoid overfitting, a dropout layer is placed between the fully connected layers.

Table 2.1 Architectures of the proposed models for tool parameter identification

Model	**1**	**2**	**3**	**4**
Input size	32×32	32×32	32×32	32×32
Stage 1	**32**-Conv3 **32**-Conv3 *Max pool*	**32**-Conv3 **32**-Conv3 *Max pool*	**32**-Conv4 **32**-Conv4 *Max pool*	**32**-Conv4 **32**-Conv4 *Max pool*
Stage 2	**64**-Conv3 **64**-Conv3 *Max pool*	**64**-Conv3 **64**-Conv3 *Max pool*	**64**-Conv4 **64**-Conv4 *Max pool*	**64**-Conv4 **64**-Conv4 *Max pool*
Stage 3		**128**-Conv3 **128**-Conv3 *Max pool*		**128**-Conv4 **128**-Conv4 *Max pool*
Last stage	FC1024 0.5-Dropout FC28	FC1024 0.5-Dropout FC28	FC1024 0.5-Dropout FC28	FC1024 0.5-Dropout FC28
No. of parameters	8,483,324	2,413,308	8,533,724	2,635,740

Model	**5**	**6**	**7**	**8**
Input size	64×64	64×64	64×64	64×64
Stage 1	**32**-Conv3 **32**-Conv3 *Max pool*	**32**-Conv3 **32**-Conv3 *Max pool*	**32**-Conv4 **32**-Conv4 *Max pool*	**32**-Conv4 **32**-Conv4 *Max pool*
Stage 2	**64**-Conv3 **64**-Conv3 *Max pool*	**64**-Conv3 **64**-Conv3 *Max pool*	**64**-Conv4 **64**-Conv4 *Max pool*	**64**-Conv4 **64**-Conv4 *Max pool*
Stage 3		**128**-Conv3 **128**-Conv3 *Max pool*		**128**-Conv4 **128**-Conv4 *Max pool*
Last stage	FC1024 0.5-Dropout FC28	FC1024 0.5-Dropout FC28	FC1024 0.5-Dropout FC28	FC1024 0.5-Dropout FC28
No. of parameters	33,649,148	8,704,764	33,699,548	8,927,196

2.3.5 Hyperparameter Setting in Deep Learning

CNN is very sensitive to the hyperparameter setting in that a small fluctuation in the hyperparameters may significantly affect its performance. The effects of the hyperparameters on the performance are diverse. Some hyperparameters have strong influence, while the others have relatively less influence. In this study, the focus is on the optimization of three influential hyperparameters, namely, learning rate lr and its decay schedule, batch size B, and number of epochs E.

The learning rate lr determines the pace of the weight update. The appropriate learning rate for each problem depends on the model and the configuration of the CNN. However, the universal rule is that high learning rates cause the system to contain an excessive amount of kinetic energy, and this causes the weight vector to chaotically explore and fail to discover the deeper and narrower parts of the loss function. Conversely, low learning rates cause the system to fail to explore wider areas of the loss function. To resolve this, a decay strategy is adopted for annealing the learning rate. This strategy reduces the rate by a constant c every fixed number of epochs ε, as shown in Eq. (2.4). This method allows substantial weight updates in the initial iterations, and low weight updates in the later stage to avoid drastic changes when the training progress reaches convergence.

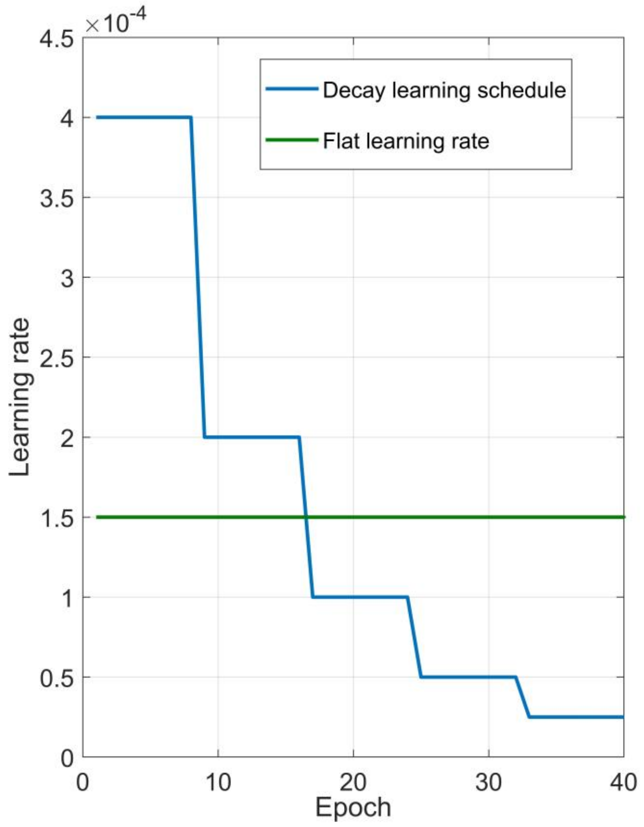


Figure 2.9 An example of step decay strategy with c = 0.5 for every 8 epochs

$$lr_e = lr_1 \times \left(\frac{1}{c}\right)^{-\left\lceil \frac{e-1}{\varepsilon} \right\rceil}, \qquad \forall e \in \{2,\ldots,E\} \tag{2.4}$$

where lr_e is the learning rate of epoch e and lr_1 is the initial learning rate. Figure 2.9 shows an example of the step decay strategy used in this study with $c = 0.5$, $\varepsilon = 8$, and $E = 40$. The number of epochs E determines the maximum number of forward-backward pass pairs of all the training data. A low number of epochs may prevent the network from exploring wider regions of the loss function. Conversely, a high number of epochs may lead to overfitting and reduce the performance. The batch size B defines the number of samples to be propagated through the network. The training set is divided into a number of batches to expedite the training process and reduce the memory allocation.

2.4 Feature Detection, Description, and Matching (FDDM) Process

Although CNN is a powerful tool for classification, it may fail to detect slightly different features of two similar tools. On the other hand, the tool recognition problem requires the system to correctly identify the exact type of the cutting tools which often have a low degree of variability between the classes. The high degree of similarity results in difficulties in distinguishing the types by mere manual observation. Therefore, the FDDM process is added to increase the accuracy.

Feature detection, description, and matching are essential components of computer vision application, especially in image recognition. This step employs the output of CNN as input information to select the candidate im_{lm} with $l \in \{1,\ldots,T\}, m \in \{1,\ldots,M\}$ to be compared with the test image t. T is the number of predictions used to select the candidates and M is the number of images in class k_n. In this study, Top-5 best predictions of CNN are used as the guide to select the reference image. Algorithm 2.2 describes the identification process that combines CNN and FDDM.

Algorithm 2.2 Identification using the combined CNN and FDDM process

```
1:   Load/capture image as test image
2:   Resize the image according to the input size
3:   Identification using CNN
4:   Record Top-5 best classes $k_l$
5:   Select the detection, description, and matching methods
6:   **while** accuracy $A_f$ < desired accuracy $A_d$
7:     **for** $l = 1 : T$
8:       **for** $m = 1 : M$
9:         Set image $im_{lm} \in k_l$ as reference image
10:        Feature detection, description, and matching
11:        **if** number of inliers/matching points $\delta \geq \lambda$
             test image $t \in k_l$
12:          **terminate for** and **while** loop
13:        **end if**
14:      **end for**
15:    **end for**
16:  **end while**
```

The feature comparison between the sample (test) image and the reference image using FDDM is conducted sequentially according to the Top-5 CNN predictions. The comparison is started against the highest probable candidate picked by the CNN. Figure 2.10 shows an example of the test image and reference images with the probability scores calculated by using the softmax function of CNN.

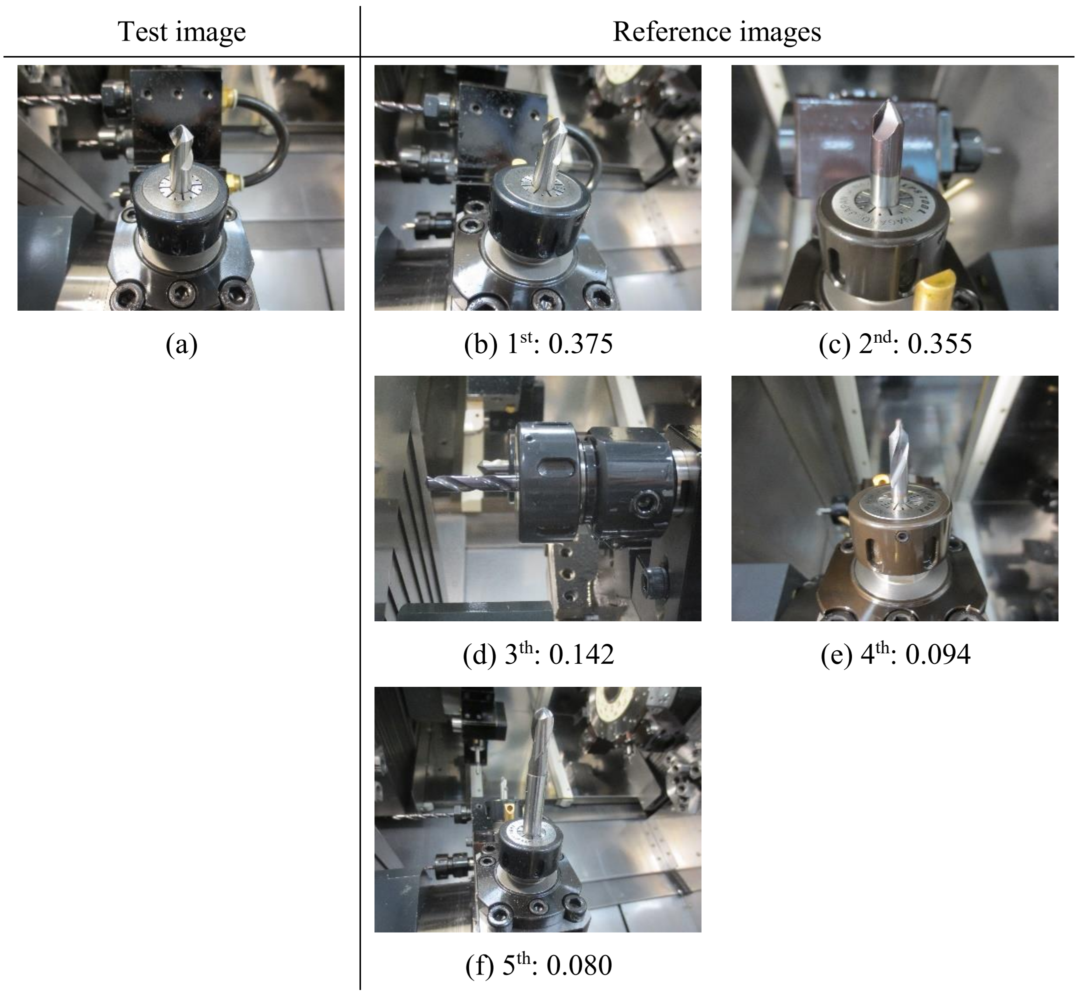


Figure 2.10 Example of test image (a) and reference images (b-f)

The feature matching was conducted between the test image (a) and the reference images (b–f), starting from the sample that has the highest CNN prediction probability (image (b)). If the number of matched points is below a threshold, then the comparison continues with the next image (c) until the last image with the least probability (f). If the matching process is unsuccessful in finding the inliers or the matching points within the pre-determined decision threshold, then the tool class is determined by the *return* strategies.

2.4.1 Feature Detection

Feature detection serves as a low-level processing step in computer vision. During the feature detection stage, each image is searched for image primitives of interest (i.e. points, lines, and regions) to highlight salient visual cues [85]. In line with the classification of visual features, the detection methods are also classified as edge, corner, and blob detectors.

Corner is defined as the intersecting point of two connected lines at which two dominant and different gradient orientations exist. Corner features are important since they may be used to locate and orient the objects, and to provide measures of their dimensions [86]. Blob is defined as a region in which the pixels inside are uniform and significantly different from the surrounding neighborhoods. Figure 2.11 shows an example of the detected corner/point features in an image of the cutting tool. The features depicted as green squares are the top 250 strongest features. As displayed in the figure, the features are mostly concentrated in the region which has two contrasting and different gradient orientations.

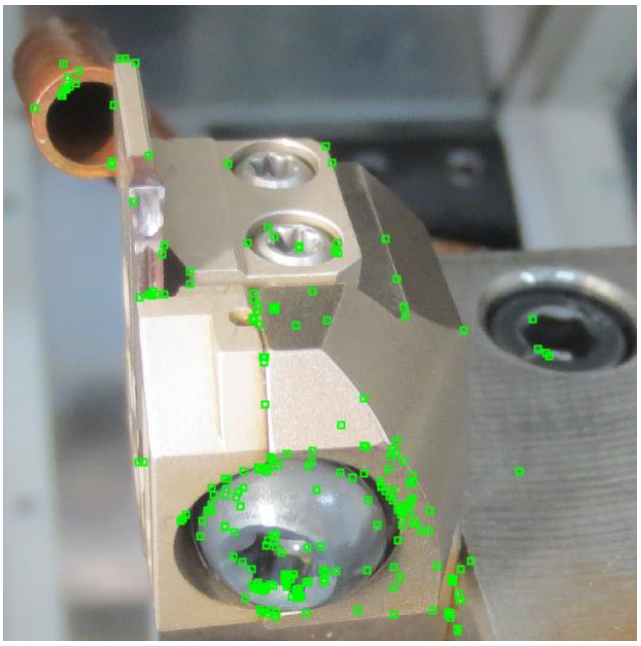


Figure 2.11 Detected point features of a cutting tool image

This process mainly focuses on the detection of corner and blob features. Methods suggested by Harris [87] and Shi-Tomasi [88], features from the accelerated segment test [89], and binary robust invariant scalable keypoints (BRISK) [90] are used for corner detection. In addition, the speedup robust feature (SURF) [91] is used as blob detector. Multiple features are obtained using two detector-descriptor pairs to reinforce the result. Double detectors are particularly useful because a single detector often fails to obtain sufficient features for the matching process.

2.4.2 Feature Description

At the feature description stage, each region around the detected key point locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors. The descriptors should be distinct and immune to changes in the viewing conditions, as well as to the errors of the detector. The description stage in this study is performed using BRISK, SURF, and Fast Retina Keypoint [92] descriptors that have previously demonstrated good performance.

The significant advantage of these descriptors is their scale and rotation invariance. These characteristics are essential because the images are captured from various positions, and the database contains a number of transformed pictures generated during the data augmentation stage. The transformation invariant descriptors are expected to perform well owing to their robustness to variations and distortions.

2.4.3 Feature Matching

Once the features of two or more images to be compared have been extracted by the descriptors, the next step involves performing feature matching. The goal of the feature matching stage is to efficiently search for likely matching candidates in the other images [93]. A good matching strategy should match most features in two images of the same object, even if the images have been altered and transformed to be distinct enough. Given a Euclidean distance metric, the matching process returns all the matches from the other images within a predetermined distance threshold τ (maximum distance). Balancing the distance threshold τ is an important aspect of this stage. Setting a high threshold may result in too many false positives (*FP*). Conversely, setting the threshold τ too low may result in the increase of false negatives (*FN*). The performance of the matching algorithm at a particular threshold can be quantified by the number of true and false matches, and failures, as defined by Fawcett [94].

TP	:	true positives, number of correct matches;
FN	:	false negatives, matches that were not correctly detected;
FP	:	false positives, proposed matches that are incorrect;
TN	:	true negatives, non-matches that were correctly rejected.

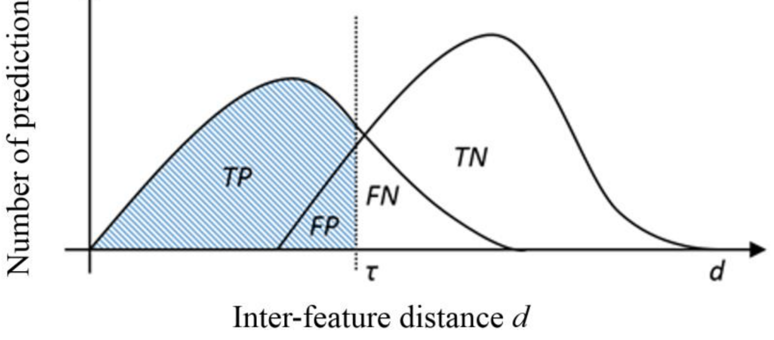


Figure 2.12 The distribution of positives (matches) and negatives (non-matches) as a function of the inter-feature distance *d*

Ideally, the true positives *TP* and the true negatives *TN* should be close to 1, while both *FN* and *FP* should be close to 0. The distribution of positives (matches) and negatives (non-matches) as a function of the inter-feature distance *d* is depicted in Fig. 2.12. As the distance threshold τ is increased, the number of true positives *TP* increases. However, at a certain point, increasing the distance threshold may increase *FP* and decrease *TP*. The highlighted area is the predicted matches, containing correct matches *TP* and incorrect proposed matches *FP*.

Once the hypothetical matches have been obtained, the next step is verifying the matches and separating the inliers from the outliers. This study implements M-estimator sample consensus (MSAC) [95], a variant of the random sampling (RANSAC) algorithm, for feature match verification and densification. This algorithm determines a geometric transform and separates the true positives *TP* from the false positives *FP*. Figure 2.13 displays the matching points of two images of a cutting tool taken from different angles. The features are detected and extracted using BRISK algorithm, after which MSAC is used to filter the matching points and remove the outliers.

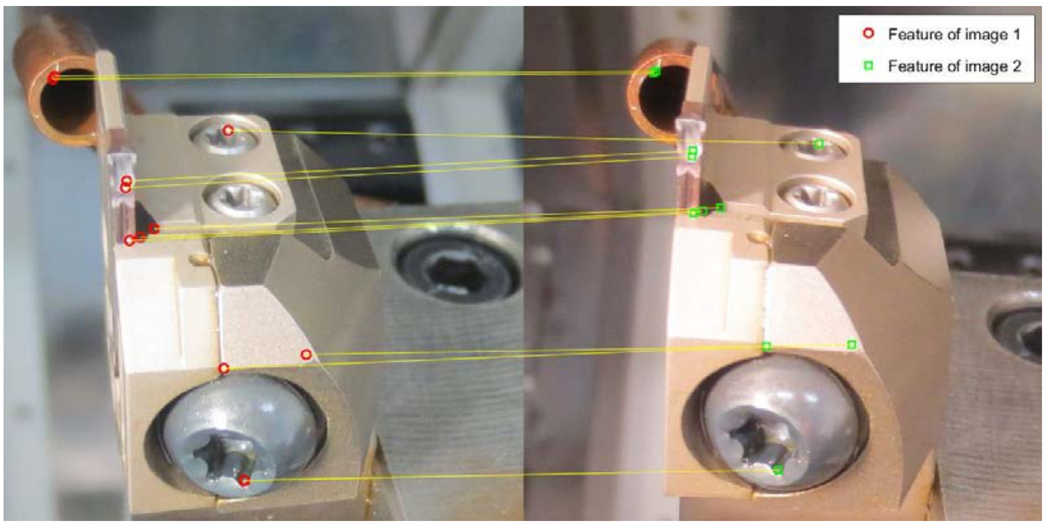


Figure 2.13 Matched feature points of two cutting tool images

Besides the distance threshold τ, another parameter that must be considered is confidence β of finding the maximum number of inliers. Both the distance threshold τ and the confidence β are trade-off parameters, influencing the accuracy and processing time. Increasing τ and decreasing β may result in an increased accuracy at the cost of increasing the processing time, and vice versa.

2.4.4 Decision Making Process

It is proposed that a decision threshold be used for deciding whether the tools in both the test and reference images are of the same type. The decision threshold λ is set on two perspectives, the number of inliers and the matching points (*MP*). As outlined in Algorithm 2.2, once the number of obtained inliers or matching points δ exceeds the decision threshold λ, the method concludes that the tool in the test image t is of the same type as the tool in the reference image im_{lm}. In this work, the combination of both the thresholds is used and is referred to as double threshold strategy. This strategy consists of two versions, inliers-*MP* and *MP*-inliers, as outlined in Algorithm 2.3. The double threshold strategy evaluates two parameters: number of matching points δ_1 and the number of inliers δ_2 in relation to their respective thresholds, which are the matching points threshold λ_1 and the inliers threshold λ_2.

Algorithm 2.3 Decision making using double threshold strategy

```
1:   if threshold strategy = inliers-MP
2:     if $\delta_1 \geq \lambda_1$
3:       if $\delta_2 \geq \lambda_2$
4:         test image $t \in kl$
5:       end if
6:     end if
7:   else if threshold strategy = MP-inliers
8:     if $\delta_2 \geq \lambda_2$
9:       if $\delta_1 \geq \lambda_1$
10:        test image $t \in kl$
11:      end if
12:    end if
13:  end if
```

Algorithm 2.4 *Default* and LMP *return* strategies

```
1:   if return strategy = default return
2:     test image $t \in k_1$
3:   else if return strategy = LMP return
4:     Calculate number of MP of all reference images $im_{lm}$, $l \in$
       $\{1, 2, ..., T\}$, $m \in \{1, 2, ..., M\}$
5:     $im_x = \max(MP_{lm})$, $l \in \{1, 2, ..., T\}$, $m \in \{1, 2, ..., M\}$
6:     $im_x \in k_x$
7:     test image $t \in k_x$
8:   end if
```

As the matching process may be unsuccessful in finding inliers or matching points within the pre-determined decision threshold λ, two mechanisms are proposed for the final prediction: using the Top-1 prediction of the CNN (default *return*) and the largest matching points (LMP *return*). Algorithm 2.4 shows both the return strategies for determining the final prediction when the matching process is inconclusive.

2.5 Object Localization Using Faster Regional-CNN

R-CNN is an extension of the CNN, and its purpose is not only to recognize objects but also locate where this object is. The location of the objects in the image is then represented using a bounding box or ROI. The ROI is identified by generating a set of proposed ROIs. Each proposed region is then tested with a process similar to that of the CNN to determine whether it corresponds to a class.

This study specifically applies the faster R-CNN [96] in order to identify the location of the tool in the image. In the original R-CNN, the proposed ROI is generated using a selective search, which frequently becomes the bottleneck during the training process. In the faster R-CNN, this step is improved, and the proposed ROI is generated based on features of the image that were already calculated with the forward pass of the CNN. This generation of the proposed region is almost cost-free. Thus, the whole identification process can be sped-up.

The faster R-CNN consists of three conceptual networks. The first is the feature network to generate the feature map from the image. The second is the region proposal network (RPN) to generate the object proposals by sliding the last shared convolutional layer over the convolutional feature map. ROI pooling is used to make the region's feature map size uniform. The third is the detection network that consists of several fully connected layers to make the prediction for each object class and bounding boxes. Figure 2.14 shows an illustration of the faster R-CNN architecture. Some examples of the object proposals are shown as green boxes and the detected ROIs are shown as yellow boxes in the figure.

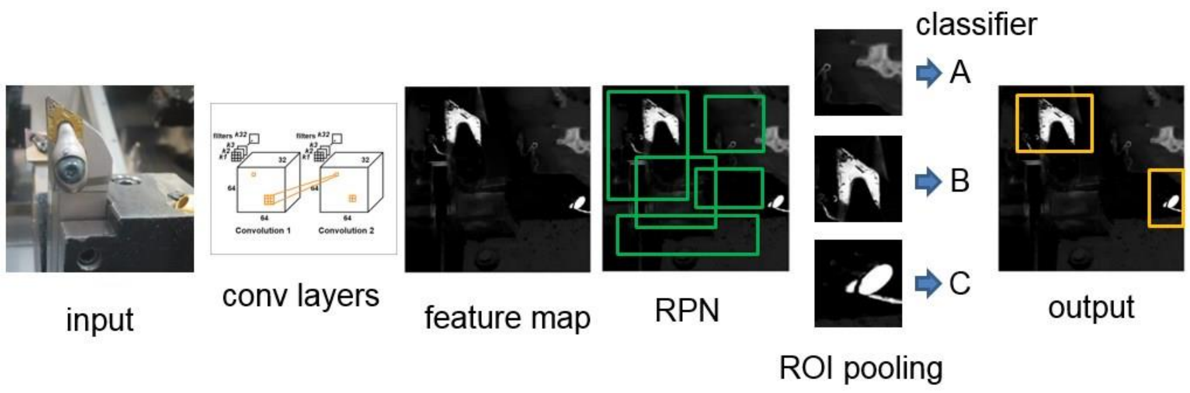
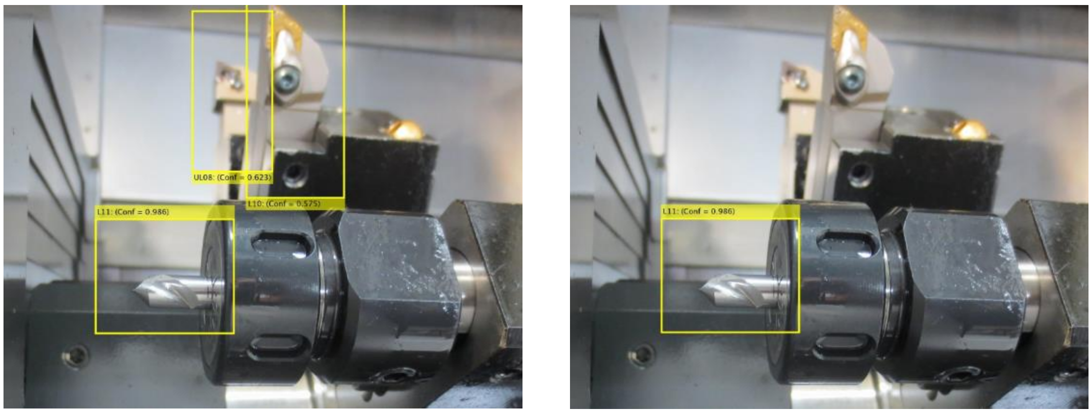


Figure 2.14 Illustration of faster R-CNN architecture

The preparation and training processes of the faster R-CNN are more complex because the developer must define the ROI of each tool of the training set images; hence it takes a long time for the data preparation. The tool localization by the faster R-CNN sometimes does not return the correct result and cannot locate the tool in the image, owing to many factors, such as noise and image quality. For the misidentification of tool type, incorrect prediction is not a problem because the aim of the R-CNN is to locate the region of interest (ROI). Moreover, the tool identification is done by using the CNN.



(a) no confidence threshold (b) confidence threshold = 0.8

Figure 2.15 Detected ROI using faster R-CNN

To overcome the inability to locate the tool (ROI) in the image, setting the correct value of the faster R-CNN confidence threshold is important. Setting the threshold value too low results in many false positives, while too high a value will result in failure of localization or false negatives. Figure 2.15 shows an example of the ROI identification using different values of confidence threshold. Figure 2.15(a) shows the ROI when no confidence threshold was applied. In this case, three ROIs are detected, including the ROI of two tools in the background. However, if the confidence threshold is set at 0.8, as shown in Fig. 2.15(b), only the ROI of the main tool in the foreground is detected.

2.6 Vision-Based Dimension Estimation

Algorithm 2.5 Automatic dimension estimation.

```
1:  Load the captured image
2:  ROI identification using RCNN
3:  if ROI is detected
4:    for $i$ = 1 : number of identified tools $I$
5:      Perform edge detection
6:      Perform estimation algorithms
7:      NC program generation
8:    end for
9:  end if
```

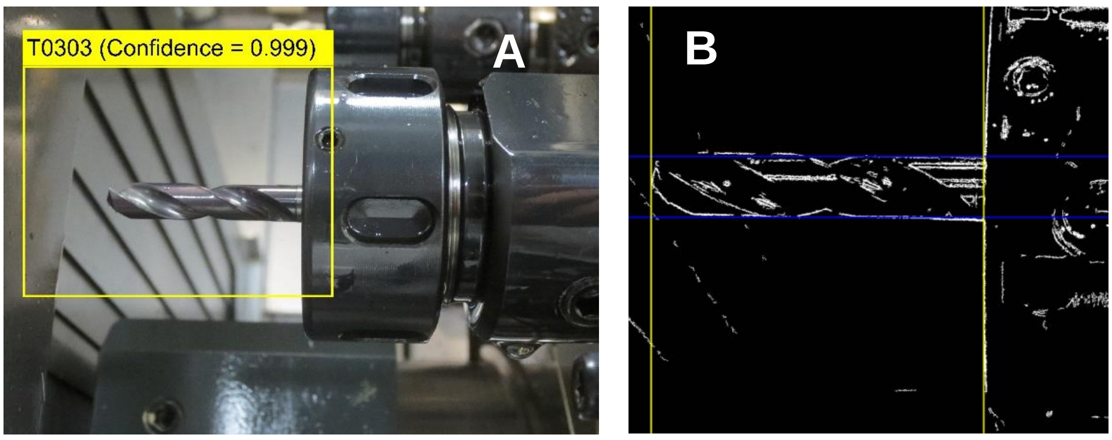
The estimation process is used to estimate the approximate dimensions that will be used to generate an NC code for the measurement process. As the image may contain unnecessary information in the background, such as the presence of a nearby tool, the estimation process is limited to the ROI that has been identified by the R-CNN.

The automatic dimension estimation involves two main processes for extracting the information from the image, namely edge detection and dimension estimation. The purpose of edge detection, as the name implies, is to identify the edges of the tool. Subsequently, estimation algorithms are used to identify the significant edges and the tool tip to avoid the noise when calculating the dimensions. The general process of automatic dimension estimation is outlined in Algorithm 2.5.

2.6.1 Edge Detection of the Identified Tool

Object localization using the R-CNN may result in multiple ROIs owing to the presence of a nearby tool in the image background. Therefore, the first step is to select the main tool that is to be identified. As this tool is located in the front, as opposed to a nearby tool that is in the background, the size of the ROI and prediction score will be much higher. After the image is cropped according to the ROI, the RGB image is converted into a grayscale format for edge detection.

The edge detection process makes use of a Canny detector [97] and Sobel operator [98] to define the edges. These methods are influenced by a sensitivity threshold wherein a higher threshold results in fewer edge pixels, and a lower threshold results in a larger number of edge pixels. The threshold must be adjusted in order to accommodate problems such as the occurrence of noise and an out-of-focus image. In this study, an adaptive threshold based on the detected edge pixels is used to adjust the threshold. In this method, the threshold is simply increased or decreased until the ratio of the detected edge pixels meets the requirement. Figure 2.16 shows an example of an R-CNN output, wherein the identified ROI is indicated by a yellow line, and the detected edge pixels of the cutting tool can be seen within the ROI boundary.



(a) detected ROI (b) detected edge

Figure 2.16 Examples of detected ROI and edge pixels of a cutting tool and its holder

The edge detection process outputs binary images. The pixel will have a value of 1 if it is located along the detected line (indicated by a white pixel) and 0 otherwise (indicated by a black pixel).

2.6.2 Tool Orientation and Direction Detection

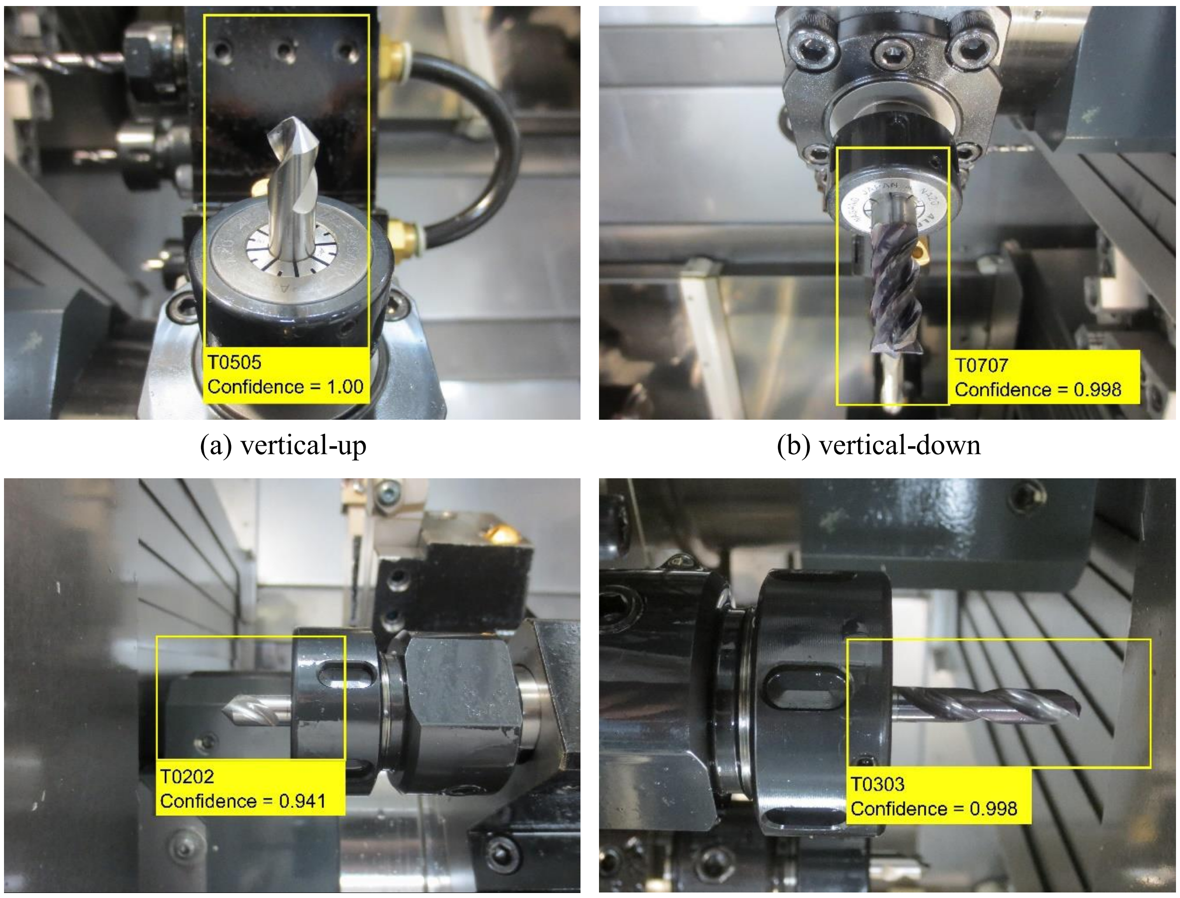
After the edge detection, a set of algorithms is developed to estimate the tool dimensions based on the obtained edge feature. Algorithm 2.6 presents the flow of the estimation method based on the detected edge pixels.

Algorithm 2.6 Flow of estimation algorithms

```
1:  Load the image after edge detection
2:  Perform orientation detection
3:  **if** detected orientation is vertical
4:     Enlarge the ROI in vertical direction
5:  **else if**
6:     Enlarge the ROI in horizontal direction
7:  **end if**
8:  Perform direction detection
9:  **for** $a = 1$ : length of image
10:    **if** density at neighborhood $N_a$ > tool tip threshold
11:       Set $N_a$ as tool tip
12:       Stop iteration
13:    **end if**
14: **end for**
15: Base identification
16: Reference line identification
17: Approximate dimension calculation
```

The first step of the estimation algorithms involves determining the tool orientation and the direction. It is necessary to identify the orientation and direction of the tool in order to identify the relative position of the tool tip and base (holder's surface). In a multi-tool multi-turret machine, there are two orientation options available based on the orientation of the milling and drilling tools, vertical and horizontal. On the other hand, for the turning tool, the orientation is strictly along the vertical direction. The relative position of the tool tip in the image is dependent on the direction of the tool. There are four options, up and down for a vertically oriented tool, and left and right for a horizontally oriented tool. Figure 2.17 shows the four orientation options and the directions of the tools.

The tool orientation and direction are detected by considering the distribution of points in the x and y directions, given that the output of the edge detection is a binary image that only has values of 0 and 1. In a noiseless image, a vertically oriented tool will be more evenly distributed row-wise and a horizontally oriented tool will be more evenly distributed column-wise. The obtained information of the tool orientation can then be used to determine the direction. The binary image is split into two zones based on the orientation, a horizontal cut for a vertical tool and a vertical cut for a horizontal tool. Subsequently, the sum of the pixel values of each zone is compared. Owing to the presence of the tool holder, the zone where the tool tip is located must have a higher pixel value than the other zone.



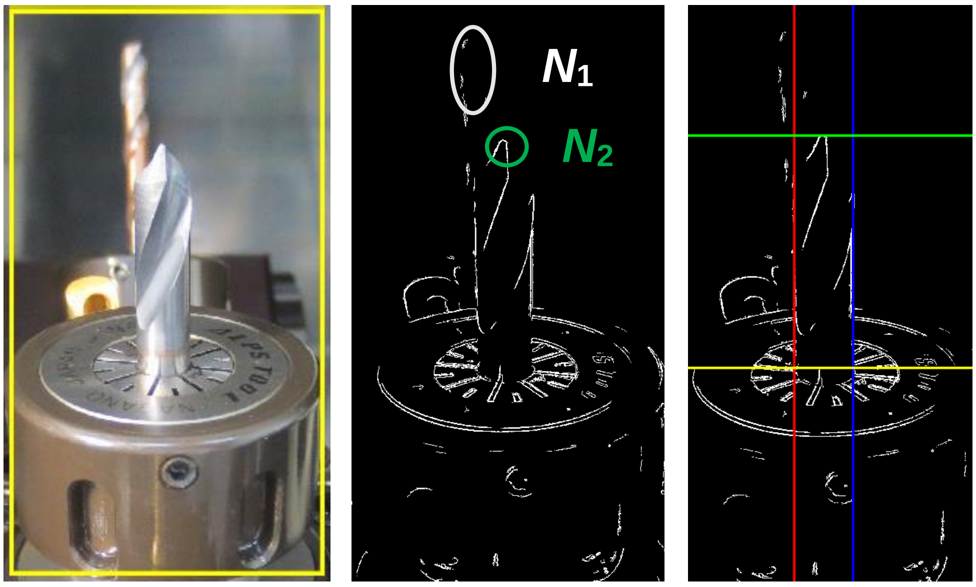
(a) vertical-up (b) vertical-down

(c) horizontal-left (d) horizontal-right

Figure 2.17 Orientation and direction of the tools

2.6.3 Tool Tip Identification

Once the relative position of the tool tip is identified after the orientation and direction detection, the next step involves identifying the exact coordinate of the tip in the image. The tool tip is identified based on a neighborhood density calculation. In this method, the density of the area around the point is calculated and analyzed. The neighborhood is distinguished as a cluster of detected points or an edge. Naturally, the tool tip forms a cluster of detected points, either in the form of clustered intersection points or an edge. In contrast, noise usually appears as isolated points without connection to the main body of the tool. Figure 2.18 shows the candidate neighborhoods of the tool tip.



(a) detected ROI (b) detected edges (c) significant features

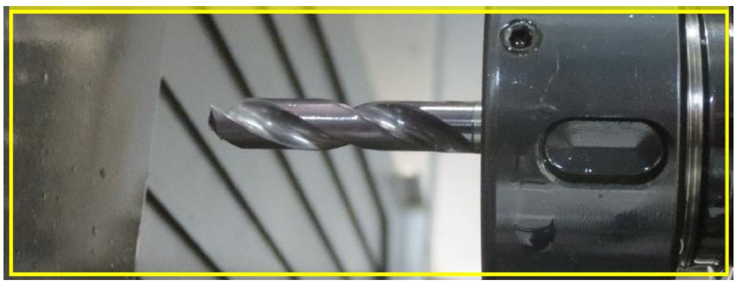
Figure 2.18 Tool tip identification based on neighborhood density

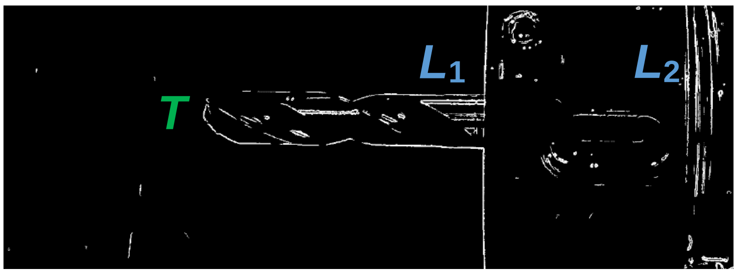
The figure shows (a) the detected ROI, (b) the detected edges, and (c) identified significant features such as edges, and tool tip. Neighborhood N_1 is formed by the noise which is the captured image of another tool, and hence, it has no connection to the other edges. Meanwhile, region N_2 is the true area that contains the tool tip.

2.6.4 Base Identification

The tool holder edge can be identified based on the orientation detection. In the case of a vertical tool, the row of the image matrix with the highest number of detected points is the holder edge, whereas, in the case of a horizontal tool, the holder edge is identified as the column with the highest number of detected points. A problem arises when the ROI covers the entire tool holder, which means there are two significant edges, i.e., both the lower and upper edges of the holder, as shown in Fig. 2.19. In this case, the base is identified as the edge nearest to the tool tip. Ideally, the two edges will have the same value of detected points. However, there is a possibility that the real base edge is weaker than the lower edge. Therefore, a threshold is set to accommodate the disparity between the two edges and to rule out noise, which can appear as other detected edges.

The figure below shows (a) the detected ROI, (b) the detected edges, and (c) identified significant edges and points. It should be noted that in figure 2.19(b), there are some significant vertical edges that could be candidates for the base. However, as line L_1 is located closer to the tool tip T compared to L_2, it can be deduced that L_1 is the true edge of the base.


(a) detected ROI



(b) detected edges

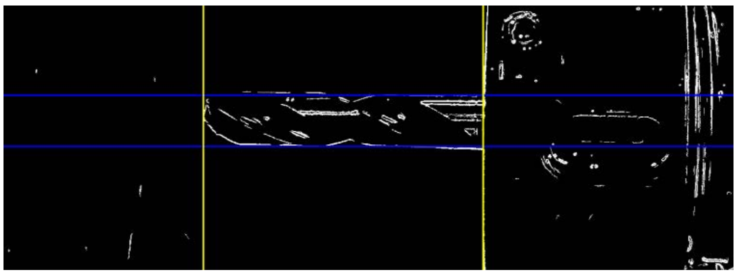

(c) significant features

Figure 2.19 Base identification

2.6.5 Reference Line Identification

The reference line is used to approximate the length per pixel. In this problem, the diameter and thickness, which can be obtained from the catalog, are used as the reference. Thus, the algorithm must be able to identify the side edge of the tool. In noiseless images, the edge can be identified easily. However, the presence of noise due to liquid, scrap, or nearby tools in the background may result in undesired detected edges. Hence, it is important to automatically identify the correct edge of the tool. To identify the correct edge of the targeted object in the image, this study uses adaptive intensity of edge detection. The adaptive process is performed by adjusting the sensitivity threshold such that the detection method is able to capture a sufficiently strong edge that can be presumed to be the significant edge while avoiding the noises.

2.6.6 Approximate Dimension Calculation

Once the reference lines have been detected, the distance between the reference lines in pixels is measured. Subsequently, the image scale (in pixels per mm) can be calculated.

Using the obtained length/height estimation in pixels and the image scale, the estimated dimensions can be calculated. However, owing to the limitations in the camera placement inside the machine, it is not possible to capture the image of some tools in exactly perpendicular position relative to the camera, which results in a capture angle. In order to accommodate this, the estimated dimension is calculated using Eq. (2.5).

$$l_a = s \times l_b (\cos(\alpha))^{-1} \tag{2.5}$$

where l_a is the estimated dimension in mm, and l_b is the estimated dimension in pixels. s is the scale of the image, which is defined in terms of pixels in mm. The capture angle α is the relative angle between the camera and the tool.

2.7 Dimension Measurement Using Displacement Sensor

Measurement and inspections are integral to maintaining the precision and quality of the machining process as well as avoiding collisions. Coordinate measuring machine is generally adopted because of its precision and flexibility with respect to a variety of products. However, it has several drawbacks, such as expensive capital cost, long operating time, maintenance burden, requirement for skillful operators, and need for an environment-controlled installation space [99]. To overcome these problems, on-machine measurement was developed as an alternative to reduce the expense and the operating time of the measurement. On the other hand, due to the demands of high precision manufacturing processes, monitoring the state of the cutting tool is required. Such a process is important to constantly detect any undesirable change in the tool dimensions that will affect the machining accuracy. An important technological element is the control of the cutting tools during the production and use in the production process [100].

Table 2.2 Advantage and disadvantage of the tool measurement method

Method	**Advantage**	**Disadvantage**
presetting	• minimizing setup time • high precision possible	• off-machine measurement • high cost equipment • long measurement time
touch-off	• on-machine • low cost equipment	• long setup time • must be adjusted constantly • not automatic setting
reference tool	• on-machine measurement • faster process than touch-off	• a passive reference tool is required (cannot be used as actual cutter) • still time consuming for setup

The tool dimension can be inspected by three methods: presetting, touch-off, and reference tool [101]. The presetting method uses the actual tool length measurement found during the setup. A special device, called as tool pre-setter, is required to set the tool length. The touch-off method is the most common method, especially in a small shop. In this method, the tool tip is moved close to the Z0 position of the part and then gently moved until it touches the feeler. In the reference tool method, as its name implies, a reference tool that has been set up is used. An actual cutter must not be used for reference because its dimension might change due to tool wear. Table 2.2 describes the benefits and disadvantages of each method.

In this study, an on-machine measurement of the cutting tool is developed by using the displacement sensor. The measurement is performed inside the machine without the need to remove the tool and the holder from the machine. The developed procedure aims to build an accurate on-machine measurement with a relatively faster process and less expensive equipment compared to the tool pre-setter method.

2.7.1 Contact-Based Displacement Sensor

In measuring tool dimensions, on-machine measurement system without removing the tool from the manufacturing instrument is desired since the measurement result can be easily fed back to the fabrication process [102]. The sensors used in the measurement are categorized into contact, non-contact, and hybrid types. The most common contact type sensors are a touch trigger and a scanning probe. On the other hand, optical sensors using a laser combined with charge coupled device are typical examples of non-contact sensors, while the hybrid type uses both contact and non-contact sensors [103].

In general, contact type sensors are able to measure objects precisely in wider ranges than non-contact types. However, the compensation of the ball radius of a touch probe and the time-consuming process for measurement are the drawbacks of the contact sensors. On the other hand, the non-contact type offers fast measurement, although it is deficient in precision compared to the touch probe. Owing to the accuracy sensitivity of the cutting-tool dimension measurement, a contact-based displacement sensor is used because it is not affected by the machining liquid, workpiece scrap, or dust as opposed to other systems such as lasers or optical instruments.

2.7.2 Automatic NC Program Generation

As the sensor position is fixed inside the machine, the tool is required to be moved until it touches the sensor head. The movements of the tool are determined by considering its type and its approximate dimensions. By utilizing the output of the estimation steps, an NC program for the measurement process is automatically generated.

2.7.3 Procedure for Measurement

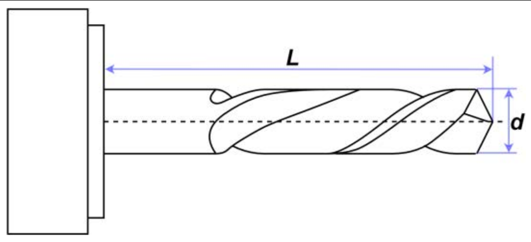
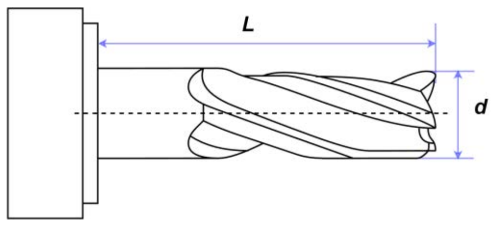
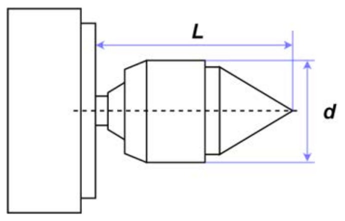
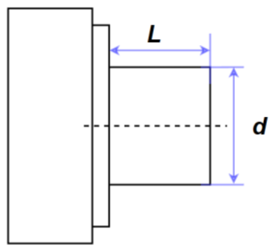
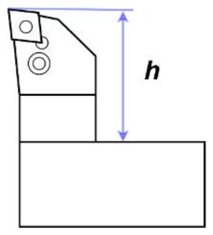
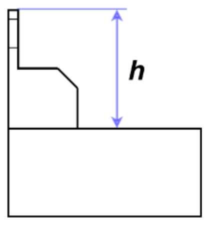
Drawing	**Tool class**	**Metric**
	Milling and drilling	Length *L* Diameter *d*
	Milling and drilling	Overhang *L* Diameter *d*
	Lathe centering	Overhang *L* Diameter *d*
	Bar stopper	Overhang *L* Diameter *d*
	Turning	Height *h*
	Turning	Height *h*

Figure 2.20 Measures of each tool type

When a tool is set, its length and diameter are regularly measured to detect the damage before machining. Length (overhang) is measured as the distance between the reference plane of the spindle and the cutting tip. The radius is measured as the distance from the center of the spindle to the cutting edge. The measurements are performed such that the tool path can be correctly offset by correcting its length (overhang) and diameter [104]. As a turning-milling machine contains a vast array of tool types, the criteria of measurement vary with each type of tool. Figure 2.20 shows the measures of each tool type.

In the case of drilling and milling tools, such as lathe centering, and bar stopper the measurement encompasses two variables, length and overhang. The measurement process for the overhang of this tool type consists of two contacts with the sensor, first the tool tip and then the reference plane, which in this case is the tool holder surface. The process flow of overhang measurement is depicted in Fig. 2.21. The movements of the tool are determined by considering its estimated dimension.

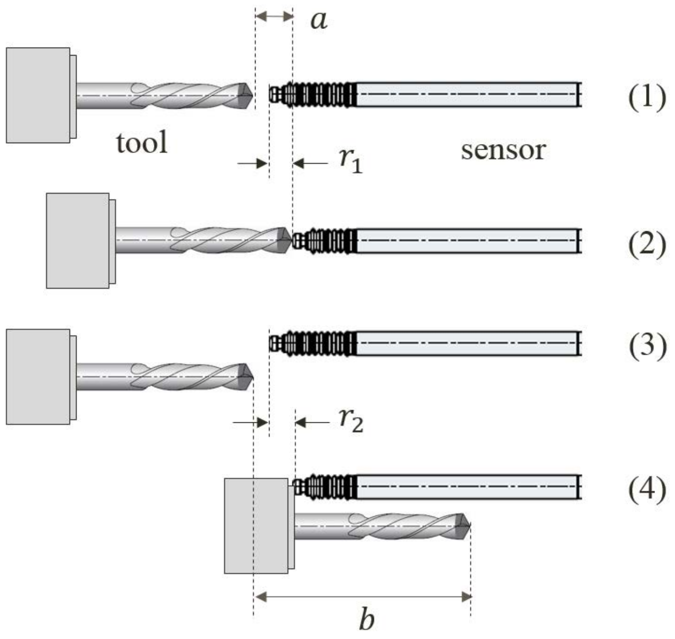


Figure 2.21 Procedure of overhang measurement

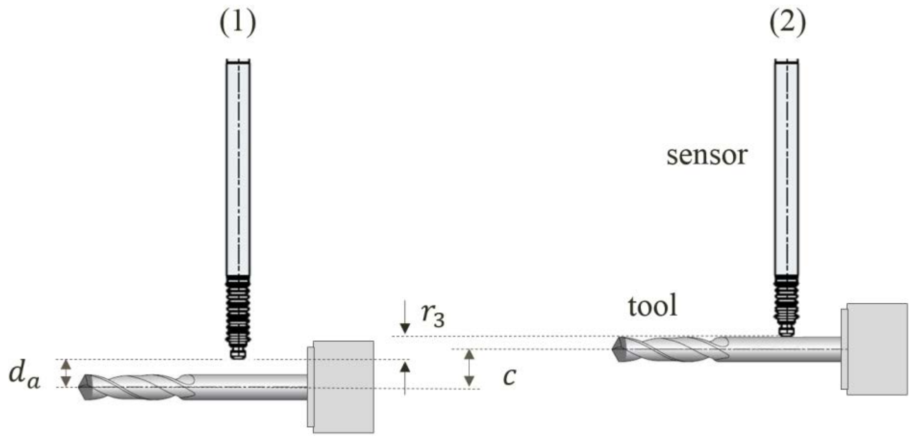


Figure 2.22 Procedure of diameter measurement

In the first step, the tool is positioned to just touch the sensor probe. Then, the tool is moved towards the sensor, pushing the sensor and resulting in the first displacement r_1. In the third step, the tool is positioned at the original position and slid down to allow the sensor to reach the holder surface. Then, the tool is moved until the sensor is pushed, resulting in a second displacement r_2. By using the generated sensor displacements data r_1 and r_2, the length of the tool can be calculated, and formulated through Eq. (2.6).

$$\begin{aligned} b &= l_a + a \\ l &= (b - r_2) - (a - r_1) \end{aligned} \tag{2.6}$$

where a and b are the first and the second movements of the tools toward the sensor. l_a is the approximate length of the tool. The diameter can be calculated similarly. The process flow of the diameter measurement is depicted in Fig. 2.22. However, it utilizes only a single contact. The formula for calculating the diameter is as follows:

$$\begin{aligned} \Delta &= c - r_3 \\ d &= d_a - 2\Delta \end{aligned} \tag{2.7}$$

where c is the movement of the tool toward the sensor and r_3 is the sensor displacement. Prior to the movement, the distance between the tool and sensor is set to d_a, which is the tool diameter data from the catalog.

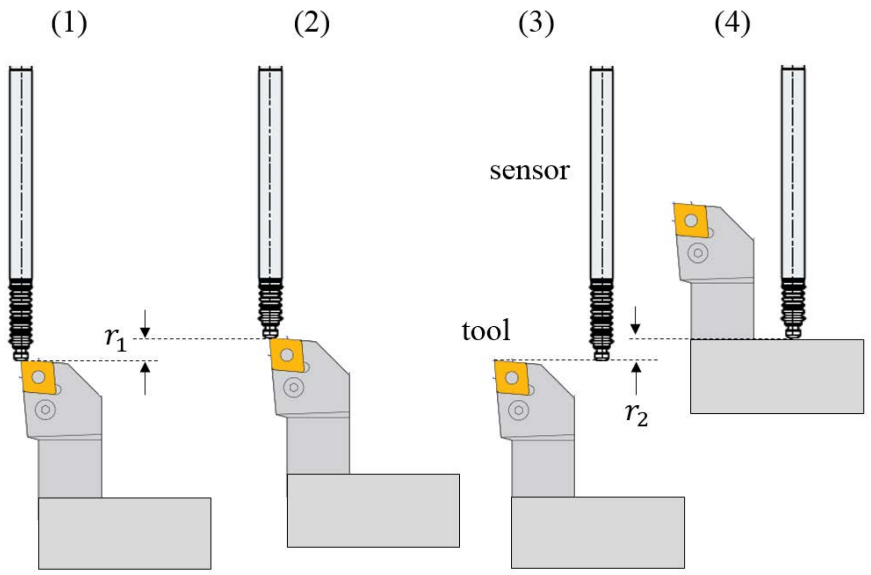


Figure 2.23 Procedure of height measurement for turning tool

The measurement process of the turning tool comprises only the height measurement, which is defined as the distance between the reference plane and the insert tip. In general, the height measurement of the turning tool is similar to the length measurement of the drilling and milling tools. The measurement process of turning tool height is described in Fig. 2.23. Since the process is similar to the overhang measurement of the milling and drilling tools, the formulation for the turning tool height calculation also follows Eq. (2.6).

2.8 Interactive User Interface for Tool Identification

The proposed methods aim to construct an automatic recognition system with minimal operator interference. However, the system still requires information from the user on both the development and prediction stages. In addition, the proposed system offers various alternatives either in the development of deep learning or in FDDM. In this case, the user may select an appropriate method in each stage that suits their problem.

Accordingly, the system is equipped with an interactive user interface which allows the user to input the information and select the appropriate method in each section. The interactive user interface program for tool identification is divided into 4 tabs corresponding to their respective sections: GUI 1 for data augmentation, GUI 2 for CNN development, GUI 3 for FDDM development, and GUI 4 for end user program. Figure 2.24 shows the framework of interactive user interface development.

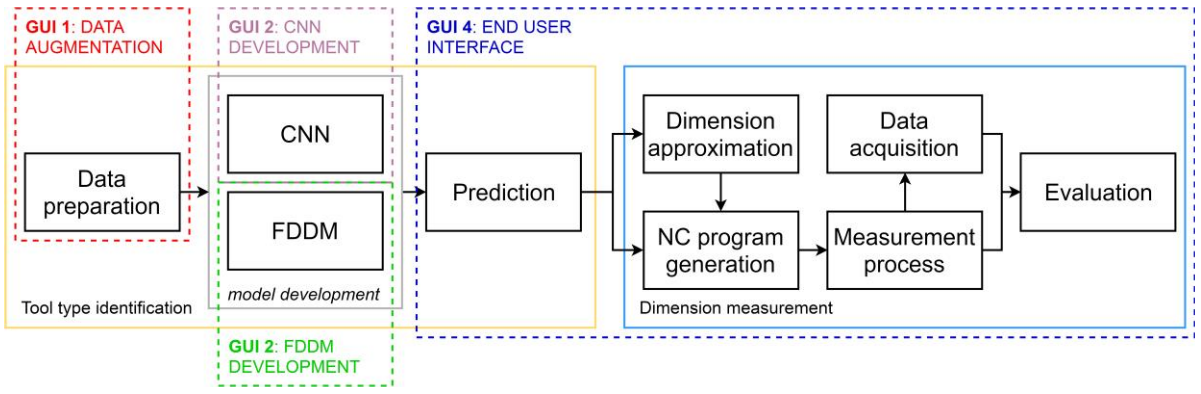
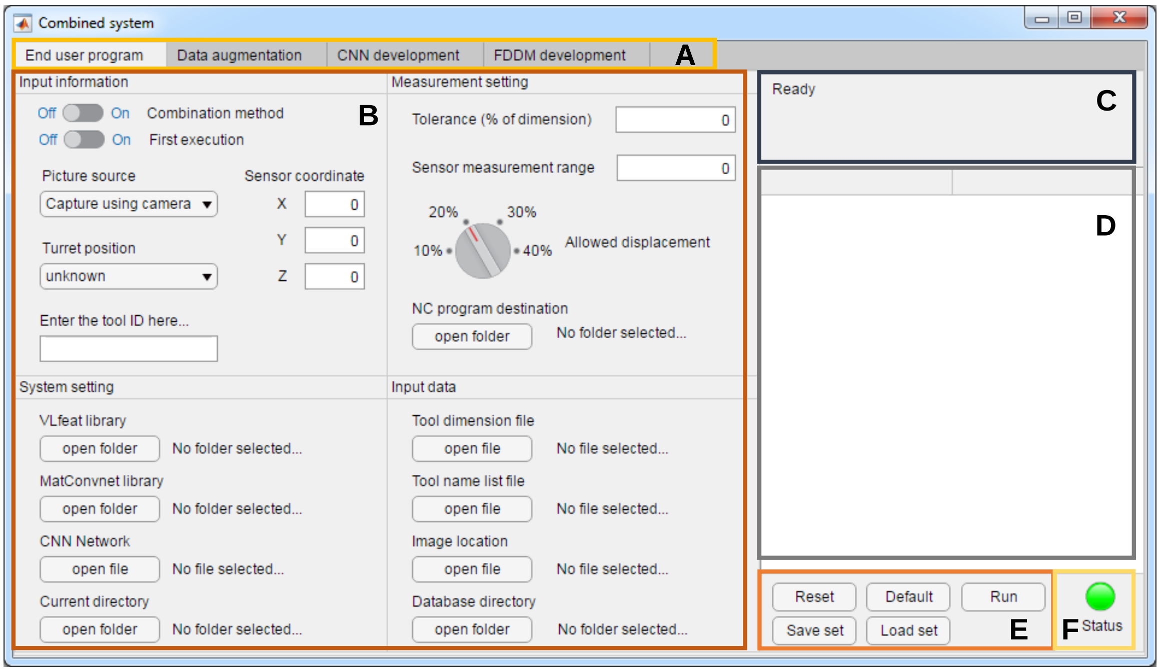


Figure 2.24 Framework of the interactive user interface development

The development of the interactive user interface uses the MATLAB graphical user interface (GUI) and the App Designer environment of MATLAB. App Designer integrates the two primary tasks of app building – laying out the visual components and programming app behavior. App Designer generates object-oriented code. Thus, this format makes it easy to share data between parts of the app. Figure 2.25 shows the design of the interactive user interface. The description of each section is explained in Table 2.3.



Off On Combination method
Off On First execution
Picture source: Capture using camera
Turret position: unknown
Enter the tool ID here...
Sensor coordinate X 0 Y 0 Z 0
Tolerance (% of dimension) 0
Sensor measurement range 0
NC program destination: No folder selected...
VLfeat library: No folder selected...
MatConvnet library: No folder selected...
CNN Network: No file selected...
Current directory: No folder selected...
Tool dimension file: No file selected...
Tool name list file: No file selected...
Image location: No file selected...
Database directory: No folder selected...
Ready

A	Tab group	**C**	Message display	**E**	Operation buttons
B	Input setting	**D**	Result table	**F**	Status lamp

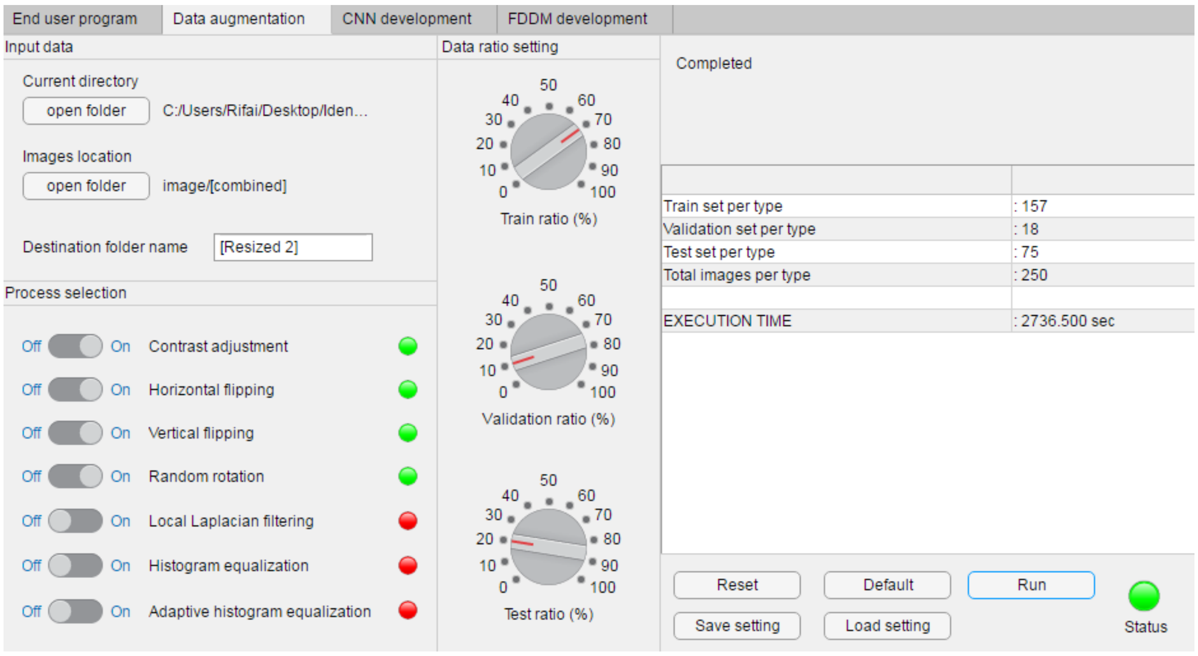
Figure 2.25 Design of the interactive user interface

Table 2.3 Description of the interactive user interface layout

No.	Section	Description
A	Tab group	The tab panels represent each section of the user interface. There are four parts presented by the four tabs: end user program, data augmentation, CNN development, and FDDM development.
B	Input setting	The input sections where the user can modify the settings, load the input data, and set the parameters. The input setting sections differ from tab to tab.
C	Message display	Display screen that shows the state of the system, instructions for the user, as well as the warning message if there is an error in the system.
D	Result table	Table that indicates the results of the process, i.e., the prediction accuracy, measured dimensions, and processing time.
E	Operation buttons	Group of buttons to start the process, reset the system, load and save the input settings.
F	Status lamp	Indicator lamp that shows the state of the application. Green indicates that the system is in the available state in which the user can modify the input settings and activate the operation buttons; red indicates that the system is running, modifications are not allowed until the running process ends or is terminated.

2.8.1 User Interface for Data Augmentation

The data augmentation tab consists of five sections: input data, process selection, ratio setting, result screen, and operational buttons. The design of this tab is shown in Fig. 2.26.



Completed

Current directory: C:/Users/Rifai/Desktop/Iden...

Images location: image/[combined]

Destination folder name: [Resized 2]

Off / On	Contrast adjustment	🟢
Off / On	Horizontal flipping	🟢
Off / On	Vertical flipping	🟢
Off / On	Random rotation	🟢
Off / On	Local Laplacian filtering	🔴
Off / On	Histogram equalization	🔴
Off / On	Adaptive histogram equalization	🔴

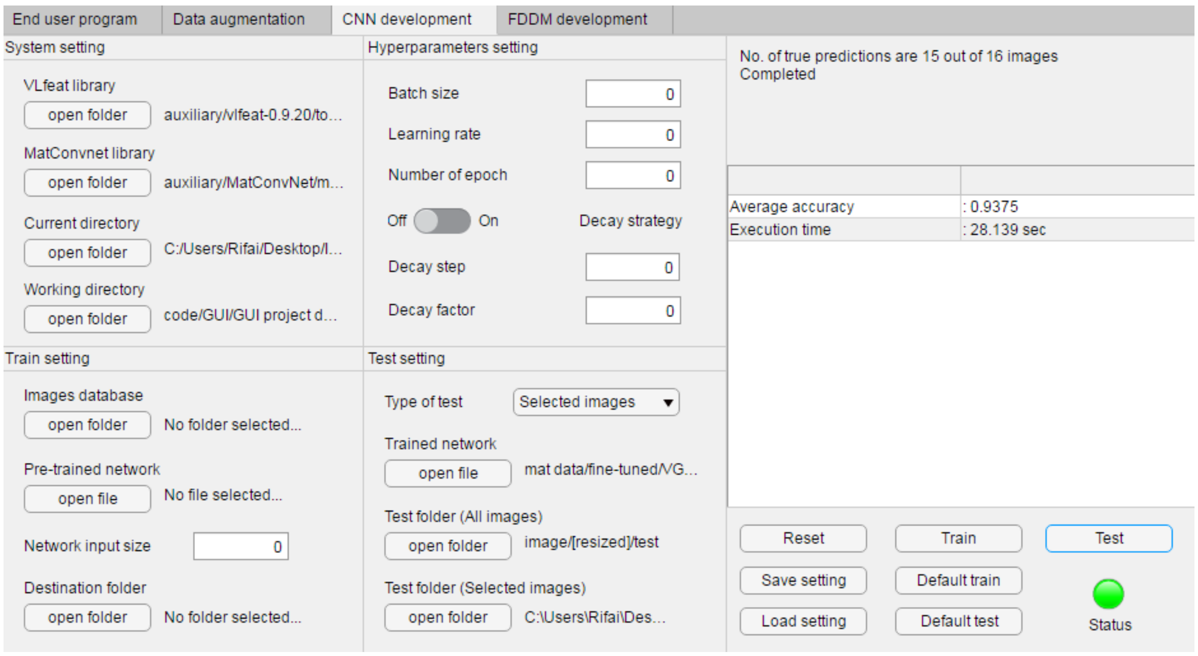
Train set per type	: 157
Validation set per type	: 18
Test set per type	: 75
Total images per type	: 250
EXECUTION TIME	: 2736.500 sec

Figure 2.26 Design of the user interface program for data augmentation process

The input data section allows the user to select the folders containing the images to be augmented. The process selection section contains seven process options that can be used for the data augmentation. These are contrast adjustment, horizontal flipping, vertical flipping, random rotation, local Laplacian filtering, histogram equalization, and adaptive histogram equalization. Once the toggle is set as “on”, (indicated by the green lamp), the respective process will be used. The data ratio setting allows the user to adjust the proportion of the newly generated images that will be used for training, validation, and testing processes. The training data set usually takes a bigger portion of the data set compared to the test and validation data sets.

2.8.2 User Interface for the CNN Development

In addition to the results and the operation buttons screens, the CNN model development tabs contain the system, train, test, and hyperparameter setting. The sections include the option to select the folders of the libraries, directories, pre-trained networks, training, and testing the images. As the name implies, the hyperparameters section allows the user to adjust the value of the batch size, learning rate, number of the epoch, as well as the decay step and factor. The user interface design for this tab is shown in Fig. 2.27.



System setting

VLfeat library
open folder auxiliary/vlfeat-0.9.20/to...

MatConvnet library
open folder auxiliary/MatConvNet/m...

Current directory
open folder C:/Users/Rifai/Desktop/I...

Working directory
open folder code/GUI/GUI project d...

Hyperparameters setting

Batch size 0

Learning rate 0

Number of epoch 0

Off On Decay strategy

Decay step 0

Decay factor 0

Train setting

Images database
open folder No folder selected...

Pre-trained network
open file No file selected...

Network input size 0

Destination folder
open folder No folder selected...

Test setting

Type of test Selected images

Trained network
open file mat data/fine-tuned/VG...

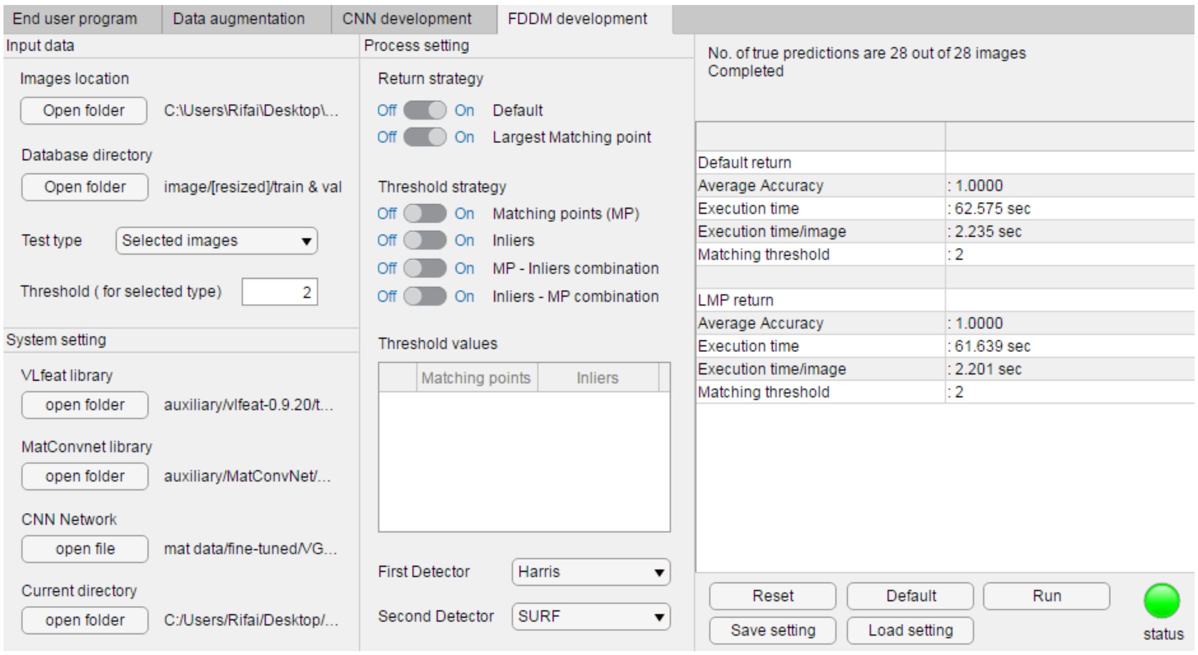
Test folder (All images)
open folder image/[resized]/test

Test folder (Selected images)
open folder C:\Users\Rifai\Des...

No. of true predictions are 15 out of 16 images
Completed

Average accuracy	: 0.9375
Execution time	: 28.139 sec

Figure 2.27 Design of the user interface program for the CNN model development



Input data

Images location
Open folder C:\Users\Rifai\Desktop\...

Database directory
Open folder image/[resized]/train & val

Test type Selected images

Threshold (for selected type) 2

System setting

VLfeat library
open folder auxiliary/vlfeat-0.9.20/t...

MatConvnet library
open folder auxiliary/MatConvNet/...

CNN Network
open file mat data/fine-tuned/VG...

Current directory
open folder C:/Users/Rifai/Desktop/...

Process setting

Return strategy
Off On Default
Off On Largest Matching point

Threshold strategy
Off On Matching points (MP)
Off On Inliers
Off On MP - Inliers combination
Off On Inliers - MP combination

Threshold values

Matching points	Inliers

First Detector Harris

Second Detector SURF

No. of true predictions are 28 out of 28 images
Completed

Default return	
Average Accuracy	: 1.0000
Execution time	: 62.575 sec
Execution time/image	: 2.235 sec
Matching threshold	: 2
LMP return	
Average Accuracy	: 1.0000
Execution time	: 61.639 sec
Execution time/image	: 2.201 sec
Matching threshold	: 2

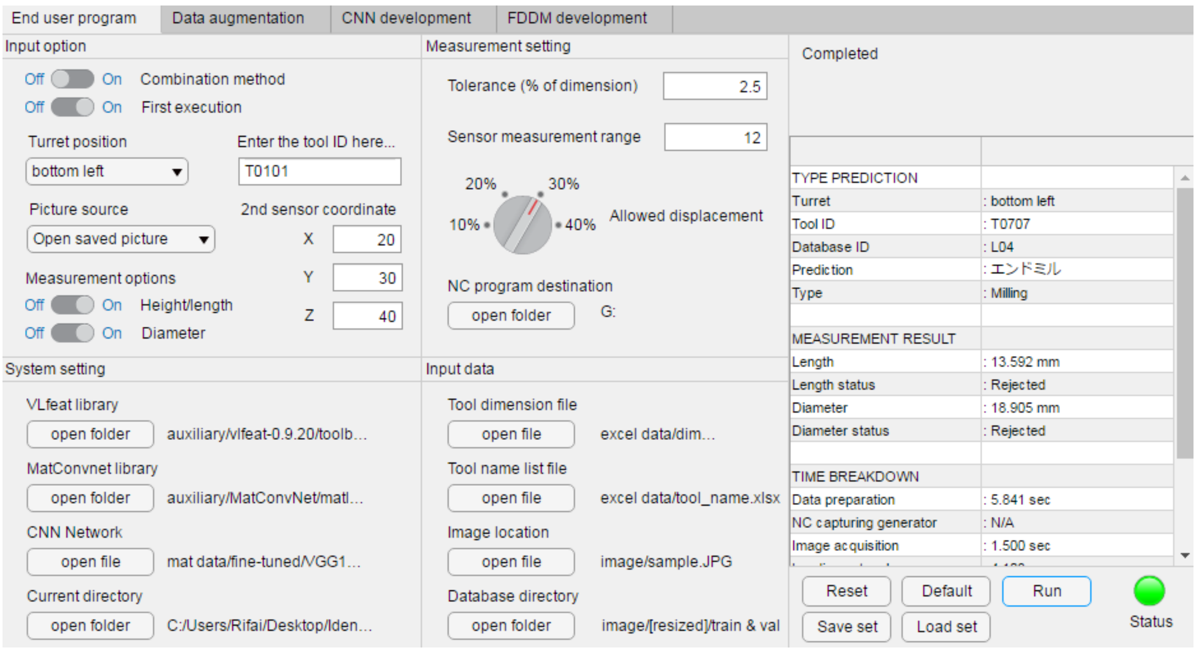
Figure 2.28 Design of the user interface program for the FDDM development

2.8.3 User Interface for FDDM Development

T The input section of the user interface for the FDDM development is divided into three parts: input data, system setting, and process setting. The system setting part, similar to the GUI of the CNN model development, requires input information of the libraries, current directory, and the trained CNN model path. The design of the FDDM development user interface is shown in Fig. 2.28.

2.8.4 User Interface for End User Program

Although the end user program is the last process in the system framework (as depicted in Fig. 2.24), it is positioned on the first tab in the user interface program. The reason is that this function serves as the finished program so that it has more interactions with user compared to the other programs which are in the development phase. Moreover, this section is designed in such way that it can be easily operated even by the users with no knowledge in computer vision, image processing, and deep learning. On the other hand, the three functions of the development phase require the users to have sufficient knowledge in their respective area.



Input option

Off On Combination method

Off On First execution

Turret position: bottom left

Enter the tool ID here...: T0101

Picture source: Open saved picture

2nd sensor coordinate: X 20, Y 30, Z 40

Measurement options

Off On Height/length

Off On Diameter

Measurement setting

Tolerance (% of dimension): 2.5

Sensor measurement range: 12

Allowed displacement: 10% 20% 30% 40%

NC program destination: G:

System setting

VLfeat library: auxiliary/vlfeat-0.9.20/toolb...

MatConvnet library: auxiliary/MatConvNet/matl...

CNN Network: mat data/fine-tuned/VGG1...

Current directory: C:/Users/Rifai/Desktop/Iden...

Input data

Tool dimension file: excel data/dim...

Tool name list file: excel data/tool_name.xlsx

Image location: image/sample.JPG

Database directory: image/[resized]/train & val

Completed

TYPE PREDICTION	
Turret	: bottom left
Tool ID	: T0707
Database ID	: L04
Prediction	: エンドミル
Type	: Milling
MEASUREMENT RESULT	
Length	: 13.592 mm
Length status	: Rejected
Diameter	: 18.905 mm
Diameter status	: Rejected
TIME BREAKDOWN	
Data preparation	: 5.841 sec
NC capturing generator	: N/A
Image acquisition	: 1.500 sec

Figure 2.29 Design of the user interface program for end user program

The input section of user interface for the end user program is divided into four parts: input option, system setting, measurement setting, and input data. The design of the end user interface is shown in Fig. 2.29. The explanation of the input information for the end user interface is described in Table 2.4.

Table 2.4 Input information in end user interface

No.	Option	Description
1	Combination method	Indicates the use of the combination threshold for FDDM prediction (*MP*-inliers and/or Inliers-*MP* combination). If set as "Off", then single threshold strategy (either *MP* or inliers threshold) is used.
2	First execution	Must be set as "On" if it is the first run after opening the program. Afterwards, the system will load the library data (VLFeat and MatConvnet libraries).
3	Turret position	Selection of the turret where the tool is placed. Four choices are available: bottom left, bottom right, upper left, and upper right. If there is no information, then select "unknown".
4	Picture source	Selection for the method to obtain the image. There are two choices: capturing using camera directly or opening the saved picture from the drives.
5	Tool ID	The input of tool ID that will be used for automatically generating the NC code for the measurement process. The data is returned in string format.
6	Measurement option	The toggle to measure the height, length, and/or diameter of the tool. If the toggle is set to "Off", the system will do only the prediction step, omitting the measurement process.
7	2nd sensor coordinate	If two sensors are used, then the user must input the coordinate of the 2nd sensor relative to the 1st sensor.
8	Tolerance	The tolerance for measurement to determine whether the measured values are still in an acceptable range compared to the dimension from catalog.
9	Sensor measurement range	The maximum displacement of the sensor in mm.
10	Allowed displacement	The allowed displacement of the sensor during the measurement process, calculated as a fraction of the sensor measurement range.
11	NC program destination	When pressed, the button displays a modal dialog box that enables the user to select the folder/drive where the generated NC programs will be saved.
12	Tool dimension file	The file (in **.xls* or **.xlsx* format) that lists the dimensions of the tools from the catalog.
13	Tool name list file	The file (in **.xls* or **.xlsx* format) that lists the type, tool ID and turret position of all installed tools.
14	Image location	Directory that contains the images to be tested. It must be supplied if selecting "Open saved picture" in picture source section.
15	Database directory	Directory which contains all the database images for comparison.

CHAPTER 3

MACHINED SURFACE QUALITY INSPECTION

3.1 Surface Roughness Estimation Problem

Real surface geometry is a complicated topic that prompted the development of various parameters to provide a good description. Gadelmawla et al. [105] listed 59 parameters, which were classified into three groups according to their functionality, namely amplitude, spacing, and hybrid parameters. Among the roughness parameters for the prediction target using vision, the majority of the previous researchers have used the amplitude parameters, more specifically, the arithmetic average roughness *Ra*. However, other indicators, such as average vertical distance from the highest peak to the lowest valley *Rz*, maximum profile peak height *Rp*, and root mean squared roughness *Rq*, are not uncommon in applications. The mathematical definition of *Ra* is as follows:

$$Ra = \frac{1}{n}\sum_{i=1}^{n}|y_i| \tag{3.1}$$

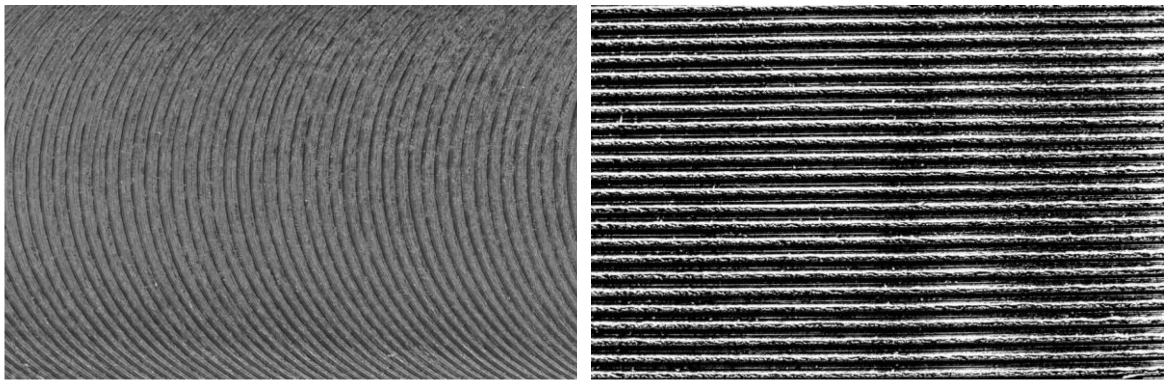

(i) pattern in milled surface (ii) pattern in turned surface

Figure 3.1 Ridge-valley patterns on machined surfaces

where y_i is the height/depth of the irregularities deviating from the mean value which is the reference line and n indicates the number of samples. In this study, we have opted to use *Ra* as an evaluation parameter. This amplitude parameter is among the most universally used due to its ease of definition and measurement, as well as its ability to provide a good general description of variations in height.

The irregularity of a machined surface caused by the machining process results in high and low spots machined into the surface. These ridges (high spots) and valleys (low spots) then define the roughness of the machined surface. In the machine vision method for roughness prediction, patterns of 'ridge-valley' are visualized. Figure 3.1 depicts the ridge-valley patterns on a machined surface.

Due to the difference in the elevations of the ridges and the valleys, the light reflection results in different light intensities on the image. Since the valley is positioned in the low spot, the light reflected from this surface is obstructed by the higher surfaces, and hence it appears darker in the image. On the contrary, the light reflected by the ridge is unobstructed, and thus it appears brighter in the image.

3.2 Unstable Cutting Mark Identification Problem

Although the issue of vibrations in machining has been covered by numerous studies in the past, it is still a very important topic that stimulates a great deal of research even at present. This persistent relevance is due to the complexity of vibration, necessitating a deep understanding and study of this phenomenon. Another factor is the negative effects of vibration, such as poor surface quality, accelerating tool wear, and unacceptable accuracy. Because vibration has a direct effect on the generated surface profile, this study aims to identify unstable cutting marks in the surface profile and simultaneously predict the surface roughness.

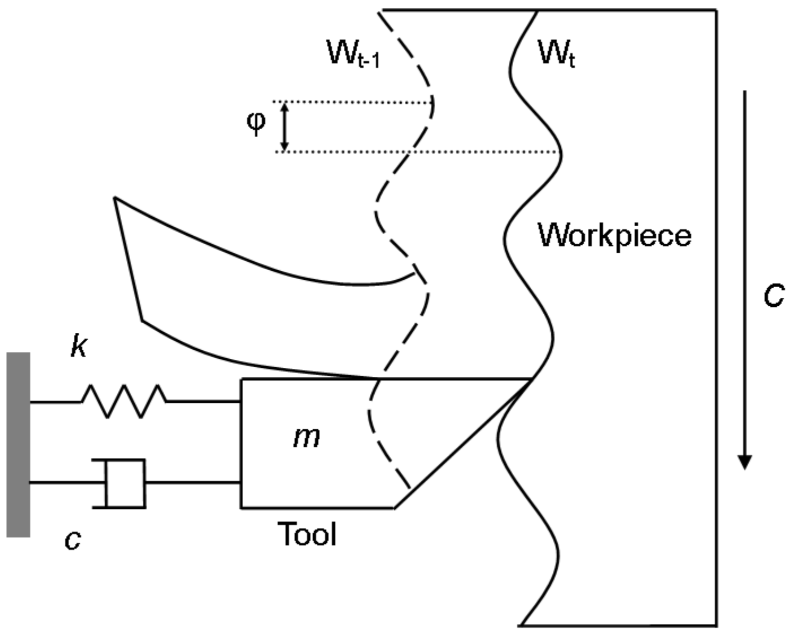


Figure 3.2 Mechanism of chatter regeneration in turning

Vibrations can be classified into two categories, primary and secondary [106]. Primary vibration is generated due to the cutting process, i.e., friction between the tool and the workpiece, thermo-mechanical effects during the chip formation, or mode coupling. Secondary vibration is due to the regeneration of waviness in the surface. By convention, the term chatter refers to the second type of vibration caused by the regenerative effect. This self-excited vibration occurs because the cutting operation involves overlapping cuts, which can be a source of vibration amplification. Figure 3.2 displays the mechanism of regeneration in a turning process.

C, *k*, *c*, and *m* denote the cutting velocity of the workpiece, stiffness, damping coefficient, and mass of the tool, respectively. W_t is the wave generated in the current revolution, while W_{t-1} is the wave from the previous revolution. The phase delay φ between the waves is the main factor that determines the generation of chatter. When the waves are in phase ($\varphi = 0$), the chip thickness variation is negligible; hence there is no surplus energy and the process remains stable. On the other hand, if $\varphi \neq 0$, the chip thickness varies, creating a dynamic cutting force at a frequency close to the natural modes, which further excite the system [107].

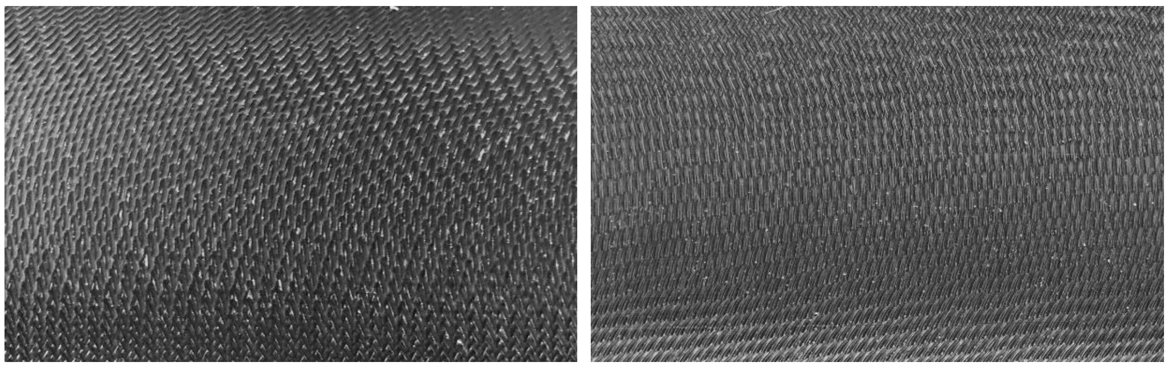


Figure 3.3 Ridge-valley patterns under unstable conditions

Surfaces shaped during the machining process when undesirable vibrations are occurring, have certain features which allow an experienced quality controller to assess their intensity [68]. In stable conditions, the ridge-valley lines left by the cutting edges forms a highly periodic pattern with a constant distance between the ridges and the valleys. Furthermore, the ridge-valley lines have uniform orientation as depicted in Fig. 3.1.

If the homogeneous pattern of the ridge-valley is disturbed, it may indicate an unstable condition during the machining. Any deviations may be a signal of surface defects, such as vibration, cracks, and breaks. While cracks and breaks usually constitute localized disturbance pattern in a specific region, the unstable cutting marks often leave heterogeneous character throughout the whole image. Figure 3.3 depicts the ridge-valley pattern under unstable conditions. In this figure, there is a clear mark that shows the discontinuity of the periodic character of the ridge-valley pattern. While the clearly visible unstable cutting marks may be easy to discern by the human eye, fine patterns might be too difficult to be recognized. In such conditions, the use of machine vision may help the operator in identifying the unstable cutting marks.

3.3 CNN for Surface Quality Inspection

This section explains the proposed vision-based surface quality inspection by developing deep convolutional neural network model for both roughness estimation and unstable cutting marks identification. These two problems belong to two different cases of prediction, namely regression problem for roughness estimation and classification problem for unstable cutting mark identification. Therefore, two networks using different loss functions are developed to accommodate the two problems. Subsequently, this study attempts to build a combined model that can be used to predict both the objectives simultaneously. Figure 3.4 shows the general framework of the development of vision-based surface quality inspection method.

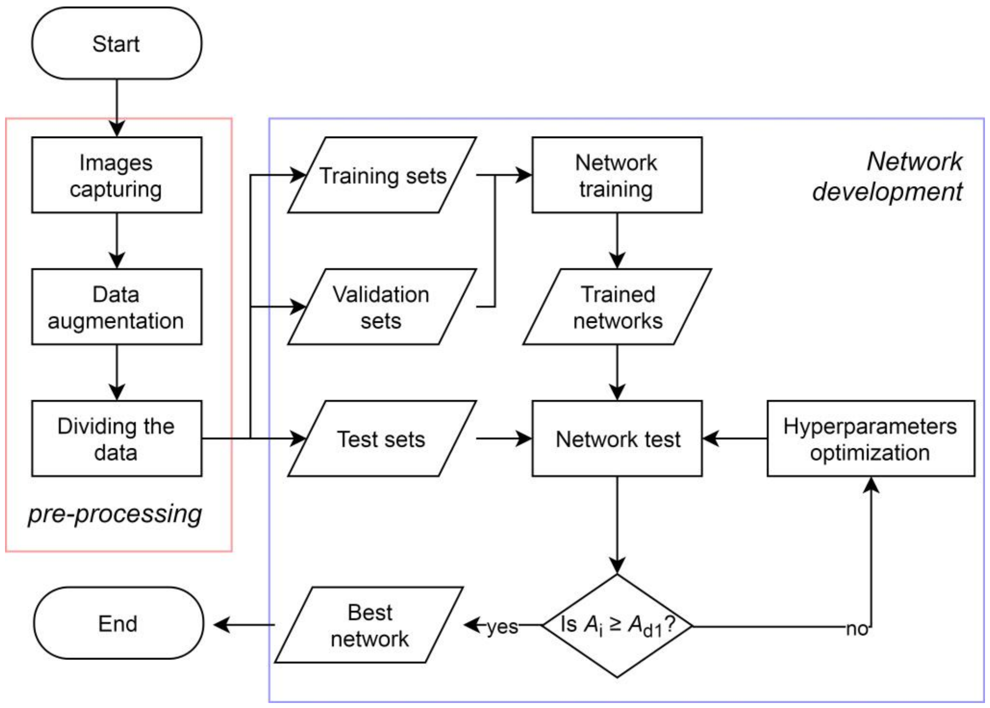


Figure 3.4 Framework of vision-based surface quality inspection method development

The system is divided into two processes. The first process is the data preparation which includes the image capturing and image pre-processing for data augmentation. The second process covers the development of CNN prediction models for surface roughness prediction, unstable cutting mark identification, or both. Unlike in the tool identification problem, transfer learning approach is not used owing to the substantial difference in the input images. While the image in the tool identification contains objects with definite and clear shapes, the surface image represents the texture which is indicated by the ridge-valley patterns. Therefore, in this case, full training approach is preferred.

3.3.1 Digital Image Pre-Processing for Enhancement

As explained in section 2.3.1, one prerequisite for building accurate deep learning models is building a large data. Models often face limitations due to the difficulty in obtaining data at such a scale. Similarly, in this study, obtaining machined surface images is an exhaustive process because only a limited number of machining experiments can be conducted. Therefore, to fulfill the requirement for big data, data augmentation is performed to multiply the images. In addition, image pre-processing is used to magnify the difference between the ridges and the valleys. All the images are in grayscale because the gradient between the valley and the ridge regions can be distinctly highlighted. Moreover, grayscale images provide a greater advantage in computational time over RGB and other color formats due to their simplicity.

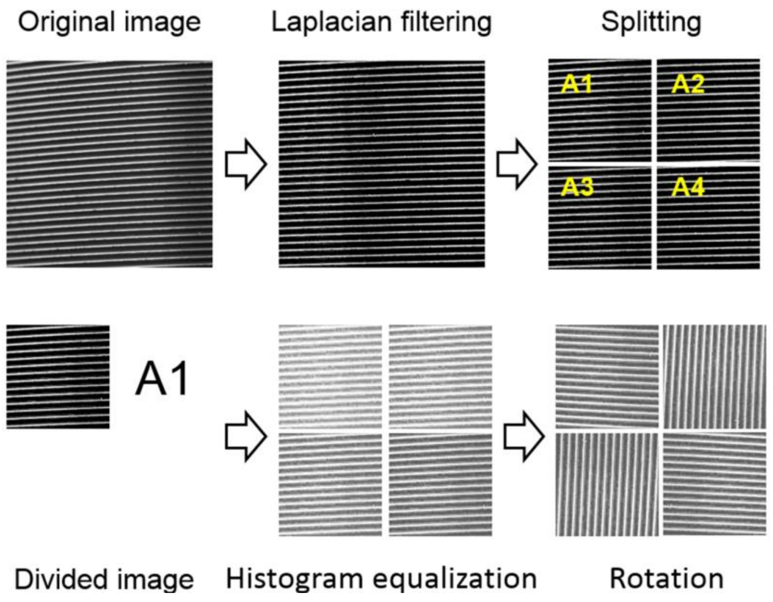
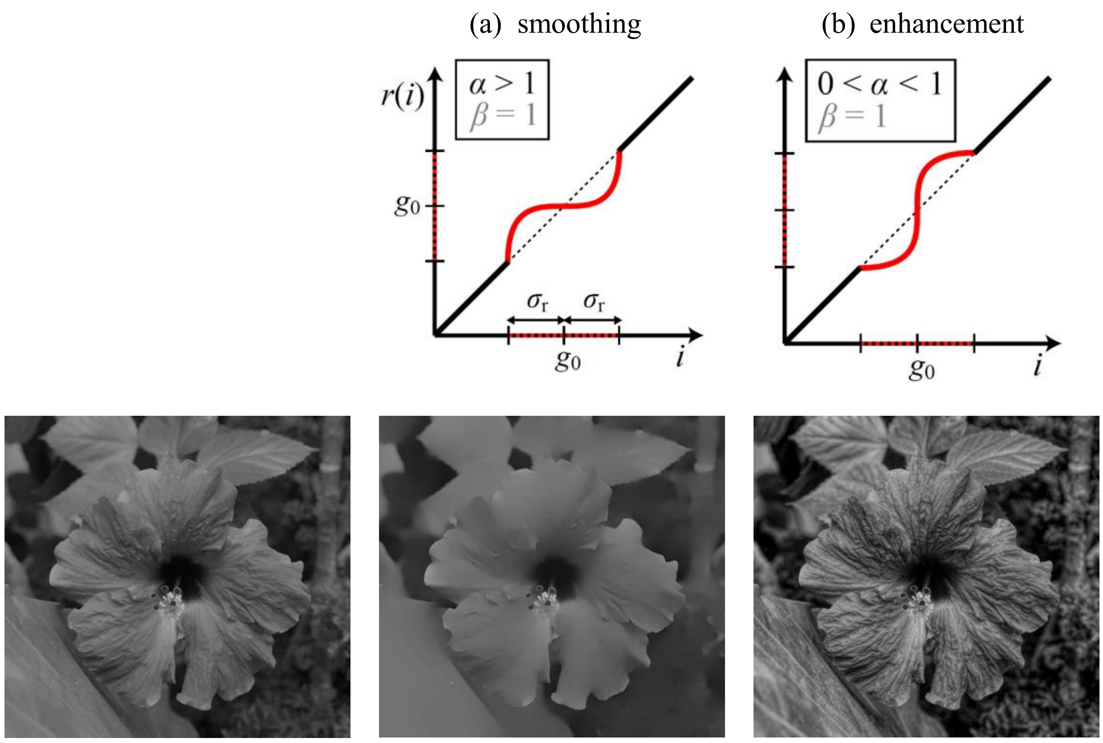


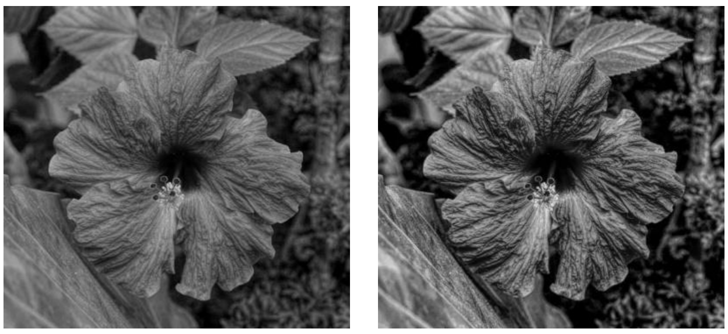
Figure 3.5 Data augmentation for machined surface images

Figure 3.5 shows an example of data augmentation for a machined surface image. Unlike the cutting tool identification which is an object identification problem, the machined surface inspection is more of pattern recognition. Therefore, the data augmentation step of this problem is also designed to enhance the texture contours of the machined surface. Two image processing methods, local Laplacian filtering and histogram equalization, are utilized both for multiplying the images and enhancing the features.



(i) input (ii) reduced details ($\alpha = 4$) (iii) increased details ($\alpha = 0.5$)

Figure 3.6 Point-wise functions (a,b), and its results (ii,iii) applied to an image (reproduced from Paris et al. [108])



(i) $\sigma_\Gamma = 0.2$ (ii) $\sigma_\Gamma = 0.5$

Figure 3.7 Effect of parameter σ_Γ applied to the same input image Fig. 3.6(i) (reproduced from Paris et al. [108])

In the first step, local Laplacian filtering [108] which is an edge-aware image processing method based on the Laplacian pyramid, is performed. These filters demonstrated high quality results for detailed manipulation and tone mapping, and particularly produced strong detailed enhancement. Figure 3.6 shows the point-wise functions for edge-aware manipulation and the results of enhancement. g_0 is the local pixel value from the Gaussian pyramid of the input and r is the remapping function. Parameter α controls the smoothing details. $\alpha<1$ increases the details of the input image, effectively enhancing the local contrast of the image. On the other hand, $\alpha>1$ smooths the details while preserving the crisp edges. Parameter β controls the tone mapping for manipulating the intensity range. Another important parameter is the amplitude of the edges σ_Γ which has a value range [0,1]. Figure 3.7 depicts the effect of parameter σ_Γ for detail enhancement.

In this problem, the above mentioned method is selected to enhance the contrast details of the surface texture image; thus smaller value of α and higher value of σ_Γ are preferred. The detail-enhancement is an important step since a detailed image contains strong patterns and features. Hence, it is easier for the CNN to identify those features during the convolutional process. Specifically, this study applies the fast local Laplacian filtering [109] which offers a faster processing while maintaining the same quality as the previous approaches.

Subsequently, the image is divided into n fragments, called sub-regions ($A1$, $A2$, etc.). In building a deep learning model, the images are resized to conform to the input size of the architecture. Resizing the whole original image will eliminate important information as the pixel numbers decrease, thus blurring the edge which represents the border of the ridge and the valley regions. To prevent information loss, the image is split before resizing so as to minimize the unnecessary simplification due to size reduction. Additionally, this step is also helpful for data augmentation since the number of images is automatically multiplied. The next step is performing histogram equalization after resizing to further improve the contrast and sharpen the edges between the ridges and the valleys. Finally, rotation is performed to accommodate the direction of the ridge-valley pattern.

3.3.2 Deep Learning Model for Roughness Estimation

3.3.2.1 Architecture of the Network

As mentioned in section 2.3.2, CNNs are trainable multistage architectures generally composed of a combination of four types of layers: convolutional, feature pooling, fully connected, and output layers. The convolutional layer is the backbone of CNN. This layer extracts feature maps of the image by executing a convolution of trainable filters/kernels over the entire input. This process is then followed by an activation function, in which a ReLU is commonly used to introduce non-linearity. After a series of convolution-ReLU layers, the image is down sampled using a feature pooling layer. Feature pooling typically involves a max or average operation to compress the feature map by forming pixel groups of a certain size, i.e., 2×2. These three layers can be stacked together to form a stage. Then, multiple stages are combined to establish powerful feature representation. The convolutional and the pooling layers are responsible for automatic feature extraction. This part distinguishes CNN from the traditional neural network or other machine learning methods in which feature extraction is performed in a separate process.

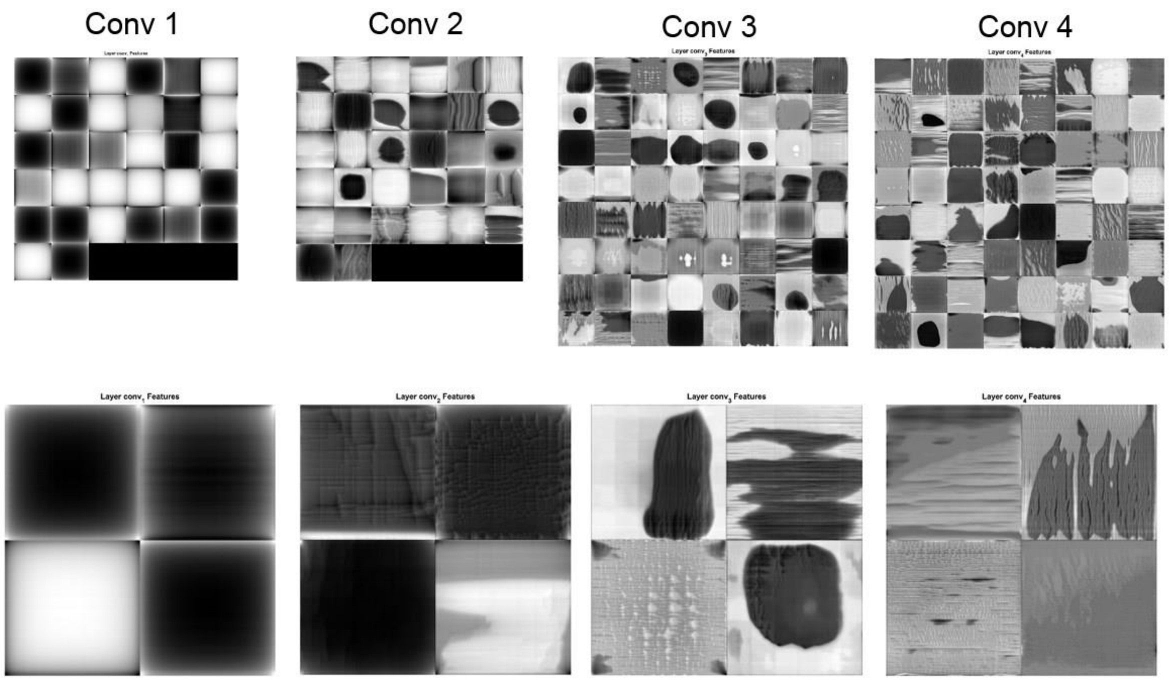


Figure 3.8 Visualization of learned features for machined surface inspection

Figure 3.8 shows the visualization of the feature maps of a machined surface evaluation model with four convolutional layers. In machined surface, grayscale images are used because the light intensity difference between the ridges and the valleys is more critical than the color. For this problem, the edge features are very important because they indicate the pattern of ridge-valley. Therefore, the feature maps also mainly indicate the edges rather than the other features.

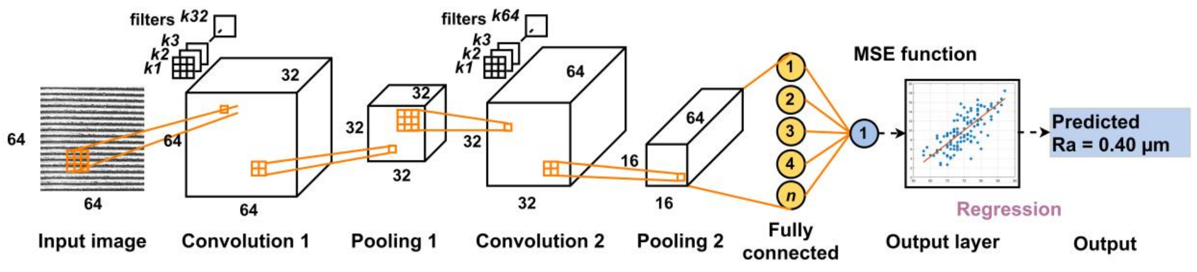


Figure 3.9 Illustration of deep learning architecture for the estimation of average surface roughness *Ra*

After a sequence of stages, the extracted features are forwarded to the fully connected (FC) layers. In an FC layer, the input feature vector is linearly converted to a new feature vector before being fed to the output function. Usually, more than one FC layer is employed. In between the FC layers, a dropout layer is often placed to avoid overfitting. The last section of the CNN is an output layer that contains a loss function. Generally, the output layer takes one of the two forms: regression or classification, based on the problem. Regression is used when the target is a numerical value, whereas classification is used when the target is a label or a class.

The output of the surface roughness estimation is a continuous numerical value for the arithmetic mean surface roughness *Ra*. Therefore, a regression output layer with a loss function is used. The loss functions used in this study are explained in detail in Section 3.3.2.2. Figure 3.9 shows an illustration of a deep learning architecture for surface roughness estimation.

Inspired by the performance of small convolutional filters as described by Simonyan and Zisserman [83], this study also utilizes a uniform 3×3 filter size as a base to design the architecture. Small convolution filters allow an increase in the network depth, which can translate into a more accurate model. The proposed architectures for surface roughness and unstable cutting evaluation are described in Table 3.1.

Table 3.1 The proposed architectures for surface quality inspection problem

Architecture	**1**	**2**	**3**	**4**
Input size	64×64	64×64	128×128	128×128
Stage 1	**32**-Conv3 **32**-Conv3 **32**-Conv3 **32**-Conv3 *Avg pool*	**32**-Conv3 **32**-Conv3 *Avg pool*	**32**-Conv3 **32**-Conv3 **32**-Conv3 **32**-Conv3 *Avg pool*	**32**-Conv3 **32**-Conv3 *Avg pool*
Stage 2	**64**-Conv3 **64**-Conv3 **64**-Conv3 **64**-Conv3 *Avg pool*	**64**-Conv3 **64**-Conv3 *Avg pool*	**64**-Conv3 **64**-Conv3 **64**-Conv3 **64**-Conv3 *Avg pool*	**64**-Conv3 **64**-Conv3 *Avg pool*
Stage 3		**128**-Conv3 **128**-Conv3 *Avg pool*	**128**-Conv3 **128**-Conv3 **128**-Conv3 **128**-Conv3 *Avg pool*	**96**-Conv3 **96**-Conv3 *Avg pool*
Stage 4				**128**-Conv3 **128**-Conv3 *Avg pool*
Last stage	FC1024 0.5-Dropout FC1	FC1024 0.5-Dropout FC1	FC1024 0.5-Dropout FC1	FC1024 0.5-Dropout FC1
No. of parameters	16,936,609	8,677,089	34,230,433	8,852,385

This study designs a network architecture to perform the surface roughness prediction based on the surface texture images. For the convolutional process, uniform 3×3 receptive fields (filter size) are used in all layers. The stride length is set to one with a padding of one in every convolutional layer for maintaining the data size. The filter number is set to 32 in the first convolutional layer, and it doubles in successive layers. Therefore, it translates to double the depth of the input in every layer. ReLU activation layers follow the convolutional layers in every stage.

Average pooling is used because it is more suitable for the texture image such as that of a machined surface. Pooling size of 2×2 with stride length set to 2 is used. This halves the width and length of the feature map in every layer. In all models, two fully connected layers are used to linearly convert the feature vector. The first fully connected layer has 1024 nodes, whereas the second layer has only one node. In between the FC layers, a dropout layer is introduced to avoid overfitting. Finally, the regression output layer with the loss function is placed at the end. The proposed model is designed to be simple and efficient so that it can be applied in a computer with standard performance GPU.

3.3.2.2 Loss Functions

As the roughness estimation is a regression problem, this study examines five regression loss functions, namely mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), log-cosh, and Huber loss. Each loss function has its characteristics, and its suitability for roughness estimation problem is investigated.

a. mean squared error (MSE)

   MSE is the most commonly used loss function in regression problem. It measures the average of the squares of the errors or the distance between the target and the predicted values. As its names implies, MSE results in a quadratic loss. The formulation of MSE and its derivative with respect to the predicted values are described in Eqs. (3.2) and (3.3), respectively.

$$MSE = \frac{1}{n}\sum_{i}\left(p_w(x^i) - q^i\right)^2 \tag{3.2}$$

$$\frac{dMSE}{dp_w(x^i)} = \frac{2}{n}\left(p_w(x^i) - q^i\right) \tag{3.3}$$

   where $p_w(x^i)$ is the predicted value of sample i, which is a function of weights w and biases b. x^i is the input vector of sample i. q^i is the target (real) value, and n is the number of samples.

b. mean absolute error (MAE)

   MAE is an average of the absolute errors or the distance between the target and the predicted values. It serves as an alternative loss function when a number of outliers are present in the data. One disadvantage of MSE is that when outliers are present, their squaring results in a very high loss even if there are only a few outliers. Thus it may skew the metric towards overestimating these outlier errors. On the other hand, MAE is a linear score which ‘weights’ the errors equally. Although the outlier errors still result in a high loss, MAE does not penalize these errors as severely as MSE does. Therefore, MAE is less susceptible to outliers than MSE. MAE and its derivative with respect to the predicted value are formulated in Eqs. (3.4) and (3.5), respectively.

$$MAE = \frac{1}{n}\sum_i \left| p_W(x^i) - q^i \right| \tag{3.4}$$

$$\frac{dMAE}{dp_W(x^i)} = \frac{1}{n}\left(\frac{p_W(x^i) - q^i}{\left| p_W(x^i) - q^i \right|} \right) \tag{3.5}$$

$$\frac{dMAE}{dp_W(x^i)} \begin{cases} 1 & if\ p_W(x^i) > q^i \\ -1 & if\ p_W(x^i) < q^i \end{cases} \tag{3.6}$$

One of the concerns in MAE is its gradient which has the same value, regardless of the error, as depicted in Eq. (3.6). Due to this, an error either small or big will result in the same loss. In addition, the second derivative is always zero at any point except at point zero ($p_W(x^i) = q^i$), which results in an undefined derivative. This condition is unfavorable for the weight updates during the training process.

c. mean absolute percentage error (MAPE)

MAPE expresses the accuracy and interprets the errors relative to the target values. MAPE and its derivative with respect to the predicted value are formulated in Eqs. (3.7) and (3.8), respectively.

$$MAPE = \frac{1}{n}\sum_i \frac{\left| p_W(x^i) - q^i \right|}{q^i} \tag{3.7}$$

$$\frac{dMAPE}{dp_W(x^i)} = \frac{1}{n}\left(\frac{p_W(x^i) - q^i}{q^i \left| p_W(x^i) - q^i \right|} \right) \tag{3.8}$$

MAPE shares the properties of MAE. There are several drawbacks in its application. MAPE cannot be applied to data that has zero target values since it will result in an infinite loss. However, this concern can be disregarded since the roughness data always has more target values than zero.

d. log-cosh

Log-cosh is a logarithmic function of the hyperbolic cosine of the errors. It has the same properties as MSE but is less sensitive to the outliers. Eq. (3.9) describes the log-cosh loss function while Eq. (3.10) shows its derivative.

$$LOG-COSH = \sum_i \log\left(\cosh\left(p_w(x^i)\right) - q^i\right) \tag{3.9}$$

$$\frac{dLOG-COSH}{dp_w(x^i)} = \tanh\left(p_w(x^i)\right) - q^i \tag{3.10}$$

e. Huber loss

Huber loss combines the properties of both MSE and MAE. The function is quadratic when the error is small ($|p_w(x^i) - q^i| \leq \delta$) and linear when the error is big ($|p_w(x^i) - q^i| > \delta$). The loss function and its derivative are formulated in Eqs. (3.11) and (3.12).

$$HUBER = \begin{cases} \frac{1}{2}\left(p_w(x^i) - q^i\right)^2 & if\left|p_w(x^i) - q^i\right| \leq \delta \\ \delta\left|p_w(x^i) - q^i\right| - \frac{1}{2}\delta^2 & if\left|p_w(x^i) - q^i\right| > \delta \end{cases} \tag{3.11}$$

$$\frac{dHUBER}{dp_w(x^i)} = \begin{cases} \left(p_w(x^i) - q^i\right) & if\left|p_w(x^i) - q^i\right| \leq \delta \\ \delta\left(\left(\frac{p_w(x^i) - q^i}{\left|p_w(x^i) - q^i\right|}\right) - q^i\right) & if\left|p_w(x^i) - q^i\right| > \delta \end{cases} \tag{3.12}$$

The deciding factor to separate the small and big errors is the cut-off hyperparameter δ, which can be tuned. As such, the challenging circumstance of Huber loss is defining the most suitable value of parameter δ since it depends on the data itself.

The comparison among the loss functions is illustrated in Fig. 3.10. Plot of MAPE is not shown since its value is relative to the target. The Huber loss in Fig 3.10 uses a cut-off hyperparameter $\delta = 2$ so that the loss in the area $-2 \leq (p_w(x^i) - q^i) \leq 2$ follows the quadratic loss. As depicted in Fig 3.10, the MSE has the steepest loss.

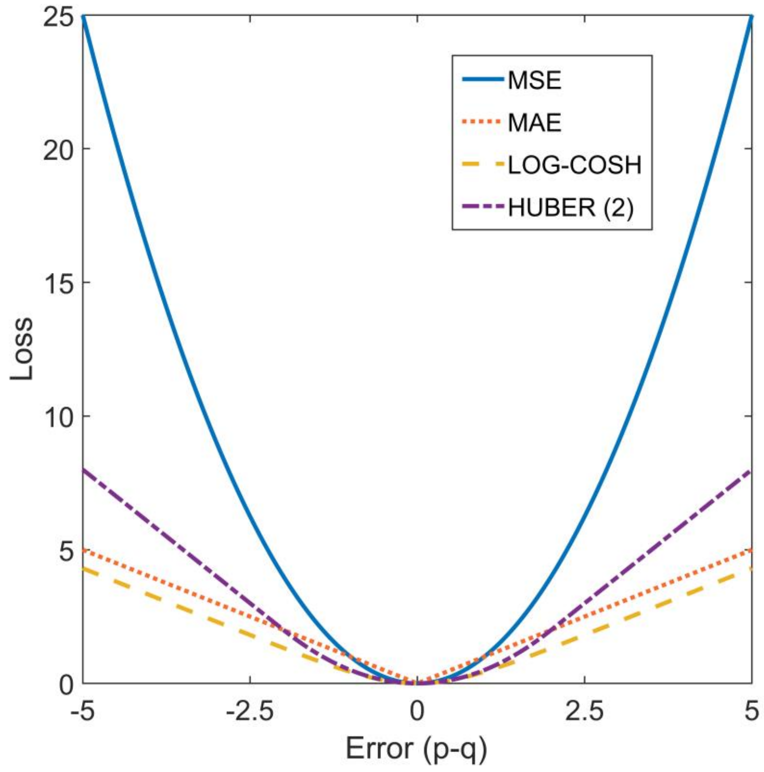


Figure 3.10 Loss vs error plot of each function

Similar to the tool identification problem, regularization is added to prevent overfitting. The regularized loss functions follow the general scheme, as formulated in Eq. (3.13). Eq. (3.14) gives an example of regularized loss function using RMSE loss.

$$L_2 = L_1 + \frac{\lambda}{2n}\sum_{w} w^2 \tag{3.13}$$

$$L_2 = \left(\frac{1}{n}\sum_{x}(p_x - q_x)^2\right)^{1/2} + \frac{\lambda}{2n}\sum_{w} w^2 \tag{3.14}$$

where L_1 is the original loss and L_2 is the regularized loss. p is the desired/target value, q is the obtained/predicted value, and n is the number of observations.

3.3.3 Procedure for Vision-Based Unstable Cutting Detection

Unstable cutting marks on a machined surface can be identified by the human eye through observation. However, this technique relies on the expertise of the technician. Moreover, it is unsuitable for a controlled production environment, and is obviously not a time- and cost-effective process. Therefore, vision-based unstable cutting mark identification has been developed as a substitute for the manual observation. Unstable cutting mark identification models use the same image pre-processing as does the surface roughness problem, for both enhancing the image and augmenting the data.

The aim of the prediction model is to classify the surfaces as stable and unstable machining specimens. Therefore, the unstable cutting mark identification problem is a classic classification problem. The models for this purpose are designed in the same way as the surface roughness estimation models. The difference lies in the last FC and the output layers. In the surface roughness estimation models, there is only 1 neuron in the last FC. However, in the unstable cutting mark identification model, there are 2 neurons, which correspond to the number of classes (unstable cutting and stable cutting). For the output layer, the regression layer is replaced with a classification layer with softmax function. The softmax layer outputs the normalized exponential of the input vector, which indicates the probability of each class. Figure 3.11 shows the deep learning architecture for the unstable cutting mark identification problem.

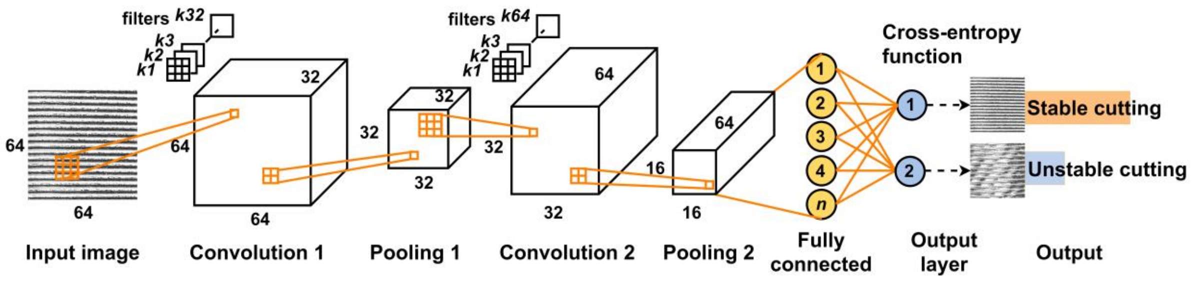


Figure 3.11 Illustration of deep learning architecture for unstable cutting mark identification

Similar to the tool type identification problems, cross-entropy is used as the loss function as described in Eq. (2.1). Hence, the regularized cross-entropy loss function for this problem follows Eq. (2.3). Accordingly, softmax function, which is used for the classification, is formulated as follows:

$$y_{iu} = \frac{e^{a_i}}{\sum_{u=1}^{C} e^{a_u}} \tag{3.15}$$

where e^{a_i} is the output of neuron i in the last layer and C is the number of classes. Softmax function takes un-normalized vector of real numbers and transforms it into normalized vector in the range (0,1), which corresponds to its probability distribution. The outputs of softmax function are suitable to be used with the cross-entropy loss for the classification task due to its probabilistic interpretation.

3.4 Combined Roughness-Unstable Cutting Mark Prediction Model

Although individual prediction models for surface roughness prediction and unstable cutting mark evaluation might offer flexibility according to the specific need, using separate models is certainly not efficient. The deep learning training is an exhaustive and time-consuming process. Furthermore, with this system, both models need to be trained separately, and it often takes a longer time because many training processes are usually performed until the best model is obtained. Therefore, developing a combined model that can simultaneously cover both the problems in a single process is seen as an alternative solution for reducing the training time. Figure 3.12 shows the last stage of the combined model architecture.

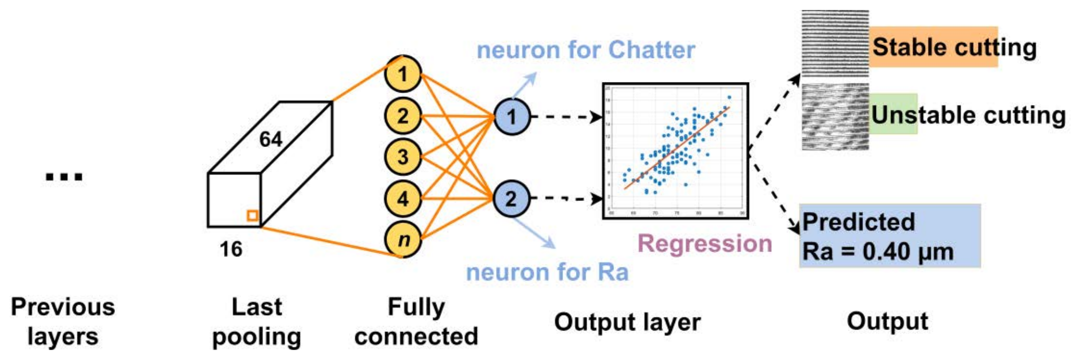


Figure 3.12 Illustration of combined surface roughness-unstable cutting mark identification model

The challenge in developing a combined model is that the surface roughness prediction and the unstable cutting mark evaluation belong to different types of problems. The former is a regression problem, while the latter is a classification problem. A straightforward combination might be unsuitable because both models use different loss functions: MSE (or other regression loss function) in surface roughness estimation, and cross-entropy in unstable cutting mark identification. Hence, a modification was performed by shifting the loss function of unstable cutting mark identification from cross-entropy to MSE (or other regression loss function). Subsequently, the labels for the training images were altered by using the value 1 for unstable cutting, and –1 for stable cutting samples. To conform to the regression problem, the new labels were treated as continuous numerical values.

As the combined model observes dual output with both deep learning and regression, both predicted outputs were expressed as continuous numerical values. Because unstable cutting marks identification was originally a classification problem, the output for this section must be converted to a discrete label. We opted to use a threshold to convert the continuous numerical value of the output into a discrete label, as shown in Eq. (3.16).

$$Ch = \begin{cases} 0 & , N_2 \leq 0 \\ 1 & , N_2 > 0 \end{cases} \tag{3.16}$$

where *Ch* is the decision variable for vibration existence, which has value 1 if the sample is unstable cutting, and 0 if it is stable cutting. N_2 is the output of the unstable cutting mark identification problem neuron in a continuous numerical value format. Since loss function must be non-negative, the formulation of MAPE and its derivative with respect to the predicted value are modified as follows:

$$MAPE = \frac{1}{n}\sum_i \left| \frac{p_w(x^i) - q^i}{q^i} \right| \tag{3.17}$$

$$\frac{dMAPE}{dp_w(x^i)} = \frac{1}{n}\left(\frac{p_w(x^i) - q^i}{|q^i| \, |p_w(x^i) - q^i|} \right) \tag{3.18}$$

The training process for roughness and unstable cutting evaluation model also follows the hyperparameter setting similar to the tool identification model. Step decay-based learning rate schedule is employed, following Eq. (2.4). Validation-based early stopping is used to prevent over-fitting and reduce the overall training time. In early stopping, the training process is stopped when there is no indication of improvement in the loss of the validation sets after a certain number of iterations.

3.5 Combined *Ra*-*Rz* Prediction

Although *Ra* can provide general overview of the roughness profile, it is less sensitive to occasional high peaks (ridges) and deep valleys. Moreover, irregularities due to defects may result in extreme outliers, which *Ra* often fails to capture. Therefore, to detect such outliers, mean roughness depth *Rz* is sometimes preferable. Two definitions of *Rz* are used, the average maximum height of the profile *Rz*, and ten-spot average roughness *RzJ*. The ten points roughness *RzJ* follows JIS-2001, and is defined as the mean distance between the five highest peaks and the five lowest valleys. The mathematical formulations of *Rz* and *RzJ* are presented in Eqs. (3.20) and (3.21). Figure 3.13 illustrates the assessed profile and explains the *Rz* and *RzJ* formulation.

$$Rt = Rp - Rv \tag{3.19}$$

$$Rz = \frac{1}{s}\sum_{i=1}^{s}\left(Rp_i - Rv_i\right) \tag{3.20}$$

$$RzJ = \frac{1}{5}\sum_{i=1}^{5}\left(Yp_i - Yv_i\right) \tag{3.21}$$

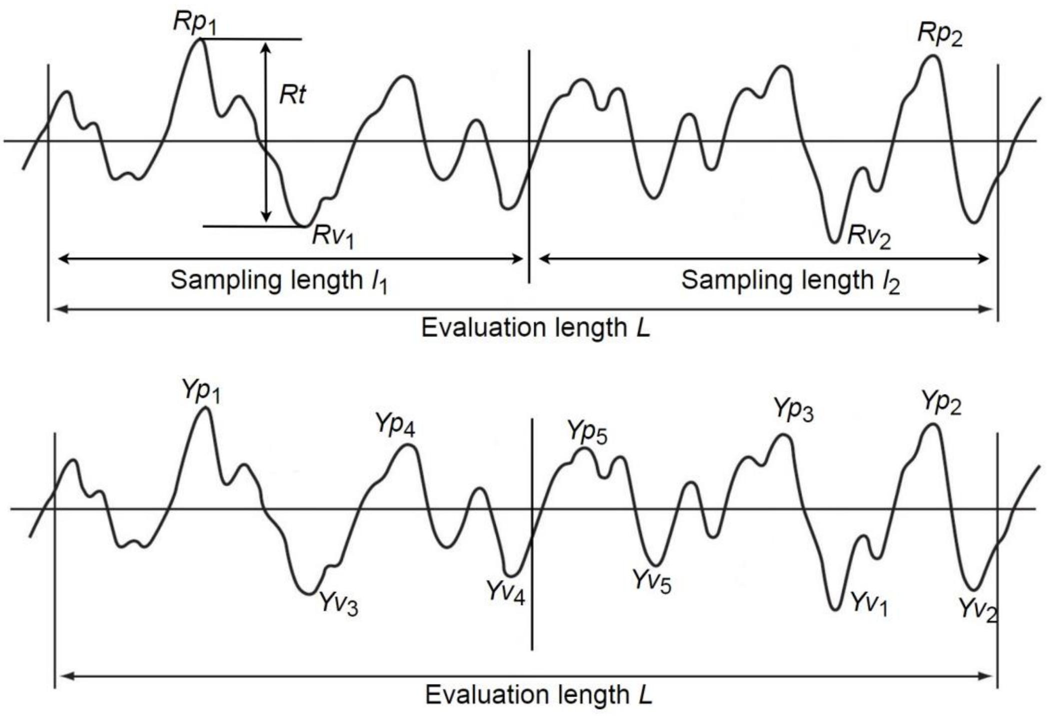


Figure 3.13 Illustration of the assessed profiles

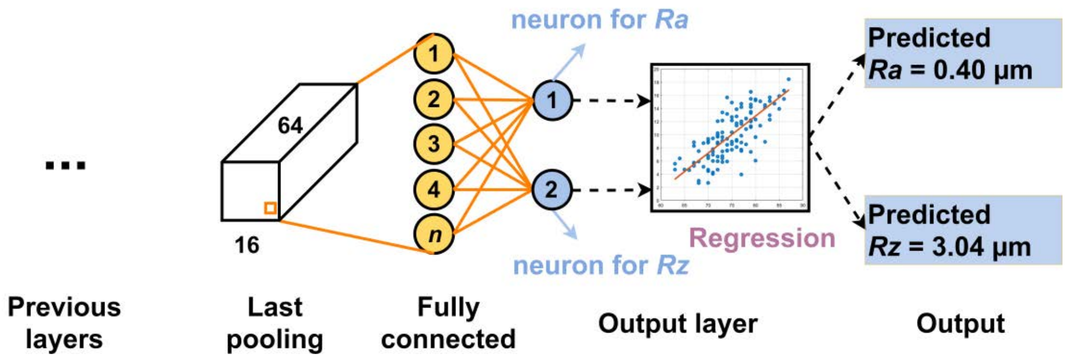


Figure 3.14 Illustration of combined *Ra-Rz/RzJ* prediction model

where *s* is the number of sampling, calculated from an evaluation length *L* divided by the sampling length *l*. Rp_i denotes the highest ridges/peaks, Rv_i denotes the lowest valley of sampling *i*, and *Rt* denotes the maximum height of the roughness. Since *Rt* is determined based on the evaluation length, it is always greater than *Rz*. However, when the sampling length *l* is same as the evaluation length *L*, then their values are equal. *Yp* denotes the five highest peaks, while *Yv* denotes the five lowest valleys of the assessed profile. To accommodate all the parameters, combined *Ra-Rz* and *Ra-RzJ* models are presented. In these models, CNN with dual output regression is used. The architecture for this case is the same as that of the single *Ra* prediction, except that the last fully connected layer has two nodes, expressing both *Ra* and *Rz/RzJ* parameters. Figure 3.14 illustrates the last stage of combined *Ra-Rz* model architecture.

3.6 Interactive User Interface for Surface Quality Inspection

Similar to the tool identification system, the surface quality inspection system is also equipped with an interactive user interface which allows the user to perform data augmentation, model training, and evaluation process both for surface roughness and unstable cutting. The interactive user interface program for surface quality inspection is divided into three tabs corresponding to their respective sections: GUI 1 for data augmentation, GUI 2 for model training, and GUI 3 for prediction. The design of this user interface follows the general format of tool identification user interface.

3.6.1 User Interface for Data Augmentation

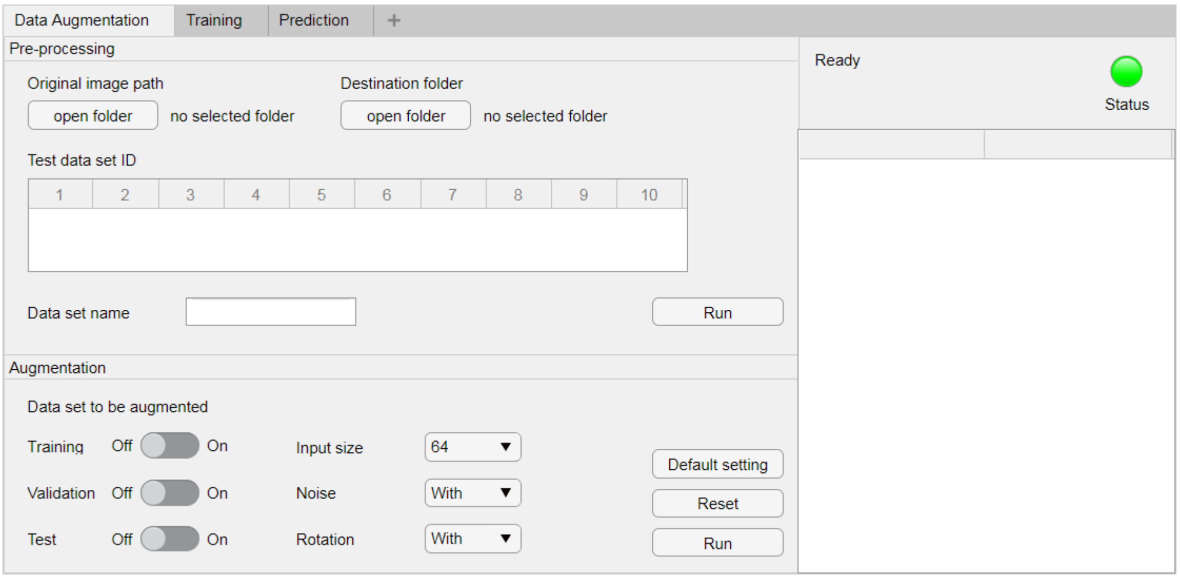


Figure 3.15 Data augmentation tab

The data augmentation tab consists of two sections, pre-processing and augmentation. The design of this tab is shown in Fig. 3.15. In the pre-processing section, the user may select the folder that contains the original images and the destination folder for the augmented images data to be saved. Next, the user is asked to input the sample ID that will be used for the test dataset. Meanwhile, the rest of the samples are automatically defined as either training or validation sets. The augmented section allows the user to select which dataset to be augmented by moving a toggle. This section also specifies the size of images, as well as the use of noise or rotation for augmentation.

3.6.2 User Interface for Model Training

Besides the operation buttons and result and message screens, the model training tab has three setting sections: data loading, hyperparameter setting, and model setting. The design of this tab is shown in Fig. 3.16.

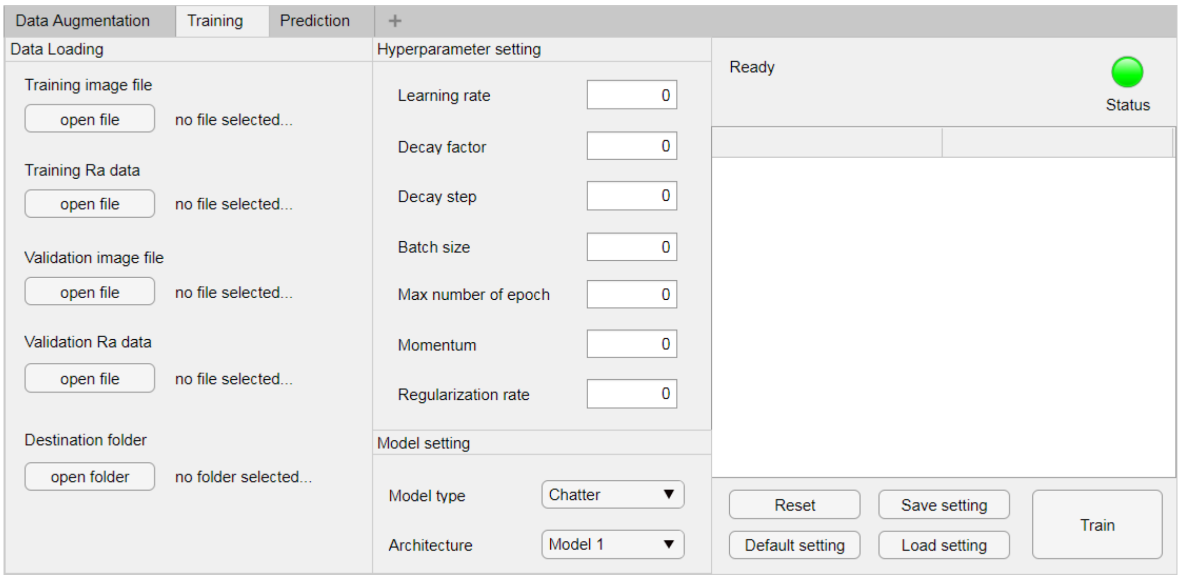


Figure 3.16 Model training tab

The data loading section allows user to select the input files for training and validation as well as the destination folder for the trained model. The input files for each training and validation set consist of the image files and the label (the target value of *Ra* and unstable cutting marks) file. There are seven hyperparameters that can be tuned by the user in the hyperparameters setting section. The model type selection consists of three options: unstable cutting marks, *Ra*, or combined evaluation. Finally, the user may select the architecture for the prediction model which corresponds to the proposed architecture for this study.

3.6.3 User Interface for Prediction

The prediction tab of the surface quality inspection user interface is comparable to end user program of the tool identification user interface. This tab allows the user to test the trained model and for making a prediction. The design of this tab is shown in Fig. 3.17.

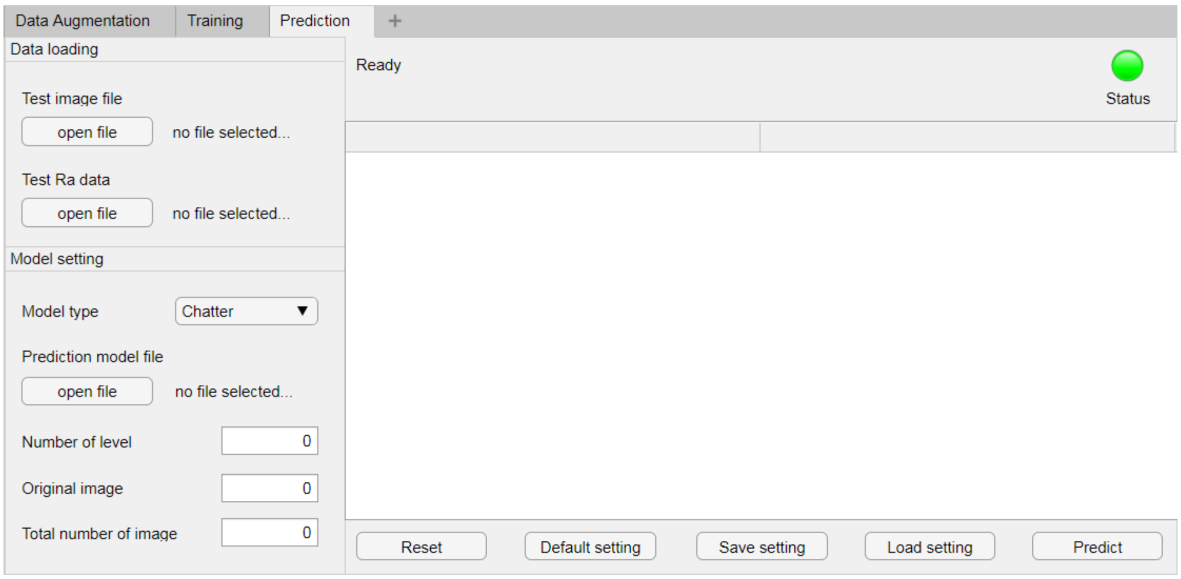


Figure 3.17 Prediction tab

The input for this tab is the test image, test label, and the trained prediction model files. User can also specify the type of the model, which can be unstable cutting, roughness, or combined evaluation model. It is also necessary for the user to input the number of original test images and the total number of samples. Results, such as the accuracy, RMSE, R^2, and prediction time will be displayed on the result screen. If the prediction process encompasses numerous samples, the detailed predicted *Ra* and the unstable cutting condition of each sample are presented in the output file.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Result of Cutting Tool Parameter Identification

The objective of the first study is to accurately predict the tool type and measure its dimensions. The performance of the tool type identification models is evaluated by comparing their accuracy and processing time. The accuracy is simply defined as the proportion of correct predictions over all the test samples. The measurement process is simply evaluated by the deviation between the estimated, measured, and catalog dimensions. Finally, the model efficiency is evaluated by considering the training, testing, dimension estimation, and measurement times.

4.1.1 Dataset and Setting

To evaluate the tool prediction methodology, a set of experiments was conducted on the dataset using the Nakamura-Tome Super NTY3 turning-milling machine with 28 tools installed in its three turrets. Figure 4.1 shows the tools used for the experiment.

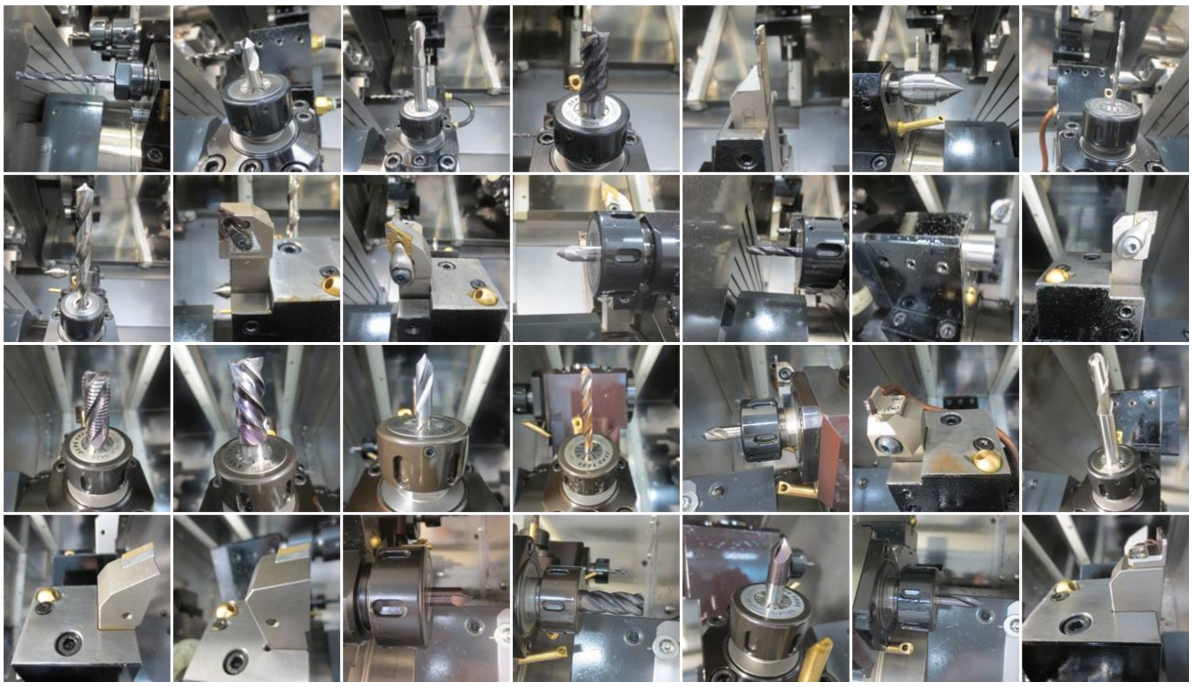


Figure 4.1 Installed tools in Nakamura-Tome turning-milling machine

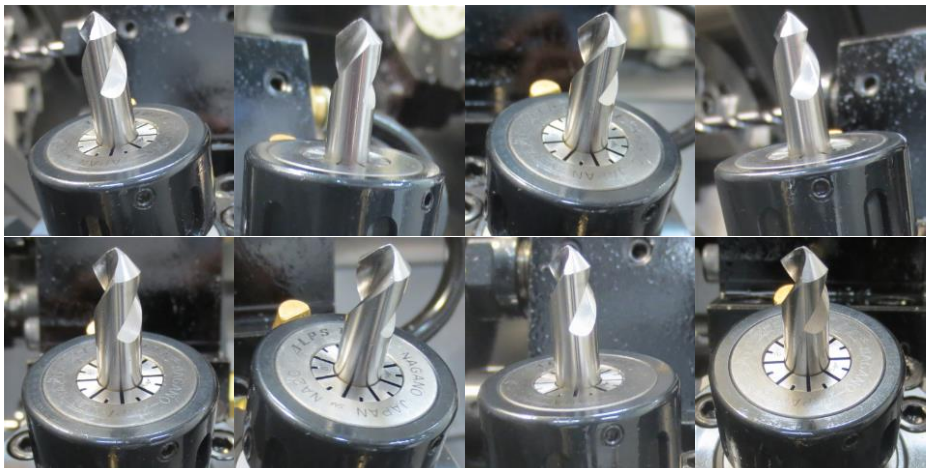


Figure 4.2 Images of a tool taken from various angles

For each tool, 50 images were captured; thus, a total of 1400 distinct images of the tools were taken from various positions and angles, as illustrated in Fig. 4.2. Some of the reasons for taking images from various angles are as follows: (i) Besides data augmentation by image processing, taking pictures from various angles is also a technique to enlarge the datasets. (ii) It is intended to build a general model that can predict the images from any angle, so that the prediction model is usable for different settings of the camera. (iii) The correct features of the tool can be learned properly by the network, which is essential for building a general model. If the same angle and the same background are used, the network might wrongly identify the features of the background as important features of the tool. Figure 4.3 gives an example of a tool and the detected features in the activation map by the convolutional layer.

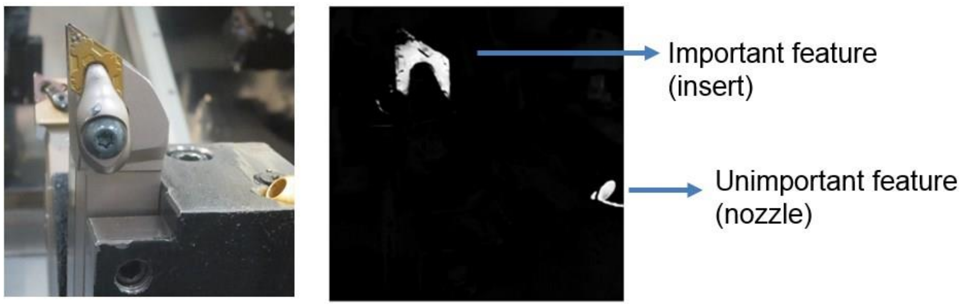


Figure 4.3 Detected features of a tool by the convolutional layer

The figure shows that there are two detected features, the shape of the insert and the nozzle. While the shape of the insert is an important feature to define the type of tool, the nozzle is not an important feature because it may have a different shape, color, or other properties in a different machine or tool block. Therefore, by taking images from various angles, the network could learn and differentiate between the important and the unimportant features of the object.

In the CNN learning process, the settings of hyperparameters were applied uniformly to ensure fair comparison between the proposed models. The learning schedule was set with two approaches: flat rate with $lr = 1 \times 10^{-4}$, and the step decay strategy with constant $c = 0.5$, and initial rate $lr_1 = 4 \times 10^{-4}$. The batch size B was set to 56.

4.1.2 Tool Identification and Localization

4.1.2.1 CNN Predictions

The training and testing processes were conducted in a personal computer with 3.50 Ghz Intel Xeon E5-1620 v3 CPU, 4 GB NVIDIA Quadro K620 GPU, and 32 GB RAM. The performance of each deep learning model was measured using Top-1 accuracy γ_1, calculated as the total number of correct predictions divided by the total number of test images, as described in Eq. (4.1).

$$\gamma_1 = \left(\sum_{n=1}^{N} \sum_{j=1}^{J} \alpha_{nj} \right) \Bigg/ \left(\sum_{n=1}^{N} J_n \right) \tag{4.1}$$

where $\alpha_{nj} \in \{0, 1\}$ has the value 1 if the prediction of the test image t_{nj} is correct and 0 otherwise. J is the number of test images t of the class $n = \{1, \ldots, N\}$. In addition, Top-5 accuracy γ_5 is also used as it is particularly useful for the FDDM method. Eq. (4.2) formulates the calculation of Top-5 accuracy.

$$\gamma_5 = \left(\sum_{n=1}^{N} \sum_{j=1}^{J} \beta_{nj} \right) \Bigg/ \left(\sum_{n=1}^{N} J_n \right) \tag{4.2}$$

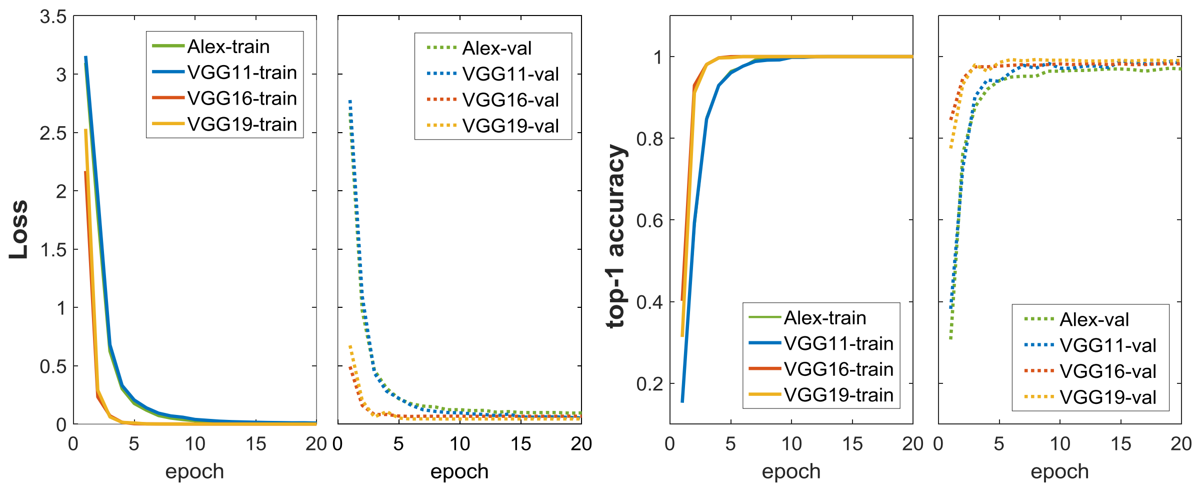
where $\beta_{nj} \in \{0, 1\}$ has the value 1 if the five highest-scored predictions of the test image t_{nj} contain the correct/actual class, and 0 otherwise. The evaluation started with the four transfer learning models. The total number of epochs E was set to 10, 20, 30, and 40 to investigate its effect on the accuracy. Table 4.1 shows the results of the four transfer learning architectures, with the highest accuracy highlighted in bold.

The prediction error decreased and accuracy increased, as the network depth increased, which is corroborated by the fact that VGG-19 achieved the highest accuracy, followed by VGG-16, VGG-11, and Alex-Net. The results in Table 4.1 indicate that the use of the step decay strategy improved the accuracy. The comparatively higher learning rate in early epochs allowed the network to explore a wider region of the loss function. As the process continued, the learning rate decreased to prevent drastic changes in the weights. Figure 4.4

depicts the change of loss and accuracy, for both the training and the validation sets, during learning process of the transfer learning models.

Table 4.1 Results of transfer learning models

Architecture	**No. of epoch *E***	**Flat rate**		**Decay strategy**		**Top-1 improvement**
		Top-1	**Top-5**	**Top-1**	**Top-5**	
Alex-Net	10	0.800	0.984	0.803	0.984	0.36%
	20	0.825	0.982	0.869	0.989	5.28%
	30	0.847	0.984	0.815	0.986	-3.79%
	40	0.893	0.994	0.911	0.992	2.00%
VGG-11	10	0.810	0.976	0.844	0.982	4.15%
	20	0.816	0.979	0.859	0.981	5.17%
	30	0.881	0.986	0.907	0.991	3.00%
	40	0.811	0.982	0.859	0.978	5.91%
VGG-16	10	0.819	0.996	0.960	0.997	17.27%
	20	0.819	0.981	0.954	0.999	16.39%
	30	0.846	0.981	0.971	0.999	14.86%
	40	0.833	0.985	0.952	0.999	14.31%
VGG-19	10	0.846	0.986	0.973	0.998	15.03%
	20	0.851	0.987	0.958	0.999	12.60%
	30	0.810	0.985	0.964	**0.999**	19.05%
	40	0.878	0.992	**0.974**	0.999	10.90%
Average improvement						**8.91%**

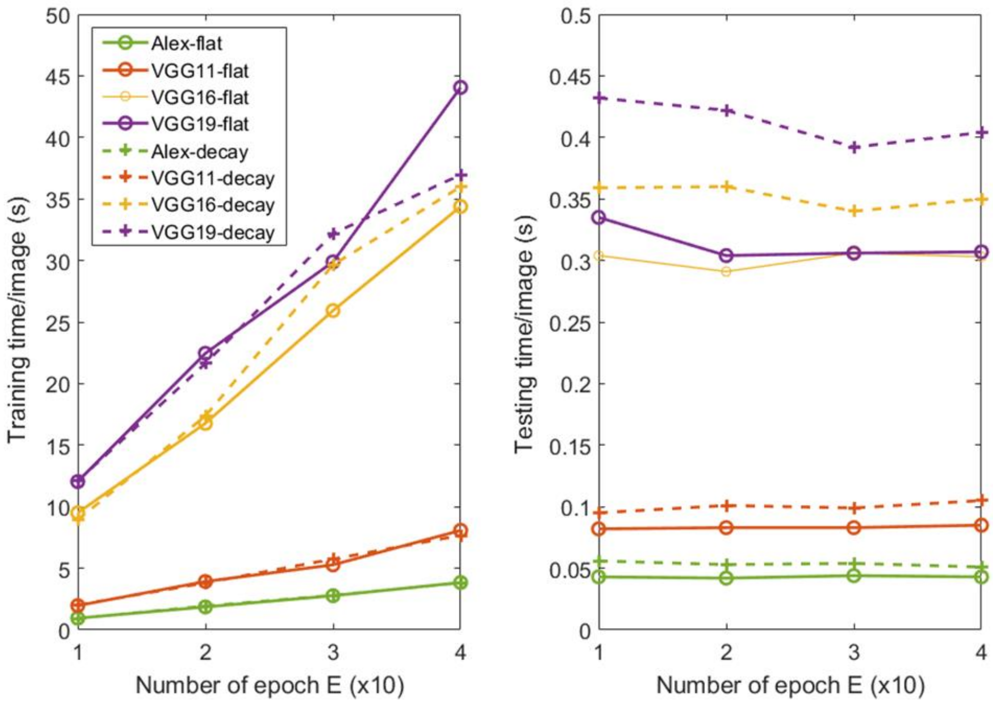


(a) loss changes (b) accuracy changes

Figure 4.4 Learning process: loss and accuracy changes

The figure shows that deeper networks converged relatively earlier than the shallow ones as indicated by the steeper changes in the loss and accuracy in the early epochs. Furthermore, the efficiency of the prediction models was examined in terms of their training time T and test time τ. The testing/prediction time was accorded a high priority, owing to its direct relationship with the machining setup time, whereas the training time was left as a secondary consideration. Figure 4.5 shows the relationship between the architectures and the hyperparameter settings with training and testing time. In this evaluation, training time T was defined as the average time required to learn one sample image in one epoch. Likewise, the test time τ was defined as the average time required to predict one test image.

The deeper network required more time for training, as shown in Fig. 4.5(a), because it involved a larger number of weight layers. VGG-19 required the longest training time in almost all the cases with various hyperparameter settings. On an average, the training time of VGG-19 was ten times as long as that of Alex-Net, five times as long as that of VGG-11, and 20% higher than that of VGG-16. Unlike the training time, the testing time was not influenced by the hyperparameter setting, and it solely depended on the architecture.



(a) Training time (b) Testing time

Figure 4.5 Training and testing time of the four transfer learning architectures with flat and decay learning schedule

Next, the evaluation of the proposed full learning models was performed. The training process of these models used the decay strategy since it was proven to perform better during the transfer learning models evaluation. The maximum number of epochs E was set at 40. Meanwhile, the values of the initial rate lr_1, constant c, and batch size B were set at the same values as those in the transfer learning model experiments. Table 4.2 shows the results of the proposed full learning models.

Table 4.2 Results of full learning models

Architecture	**Top-1**	**Top-5**	**Training time/ sample T [s]**	**Test time/ sample τ [s]**
Model 1	0.925	0.990	0.0014	0.0008
Model 2	0.931	0.986	0.0017	0.0006
Model 3	0.931	0.986	0.0013	0.0008
Model 4	0.921	0.984	0.0017	0.0006
Model 5	0.938	**0.994**	0.0066	0.0023
Model 6	0.938	0.986	0.0053	0.0038
Model 7	0.938	0.993	0.0102	0.0024
Model 8	**0.944**	0.989	0.0108	0.0043

The results indicate that some of the proposed models performed favorably. Despite having smaller input sizes and narrower architectures, all the proposed models were able to outperform Alex-Net and VGG-11. The highest Top-1 accuracy was recorded using model 8 with an accuracy of 0.944. Meanwhile, the deeper architecture of VGG-16 and VGG-19 guaranteed a higher accuracy as compared to the other models. The training time was affected by the input size, depth of network, and, to some extent, the number of parameters. Nevertheless, the lighter configurations (with smaller inputs and narrower network) of the proposed models ensured a significantly faster computation time for both the training and the testing processes.

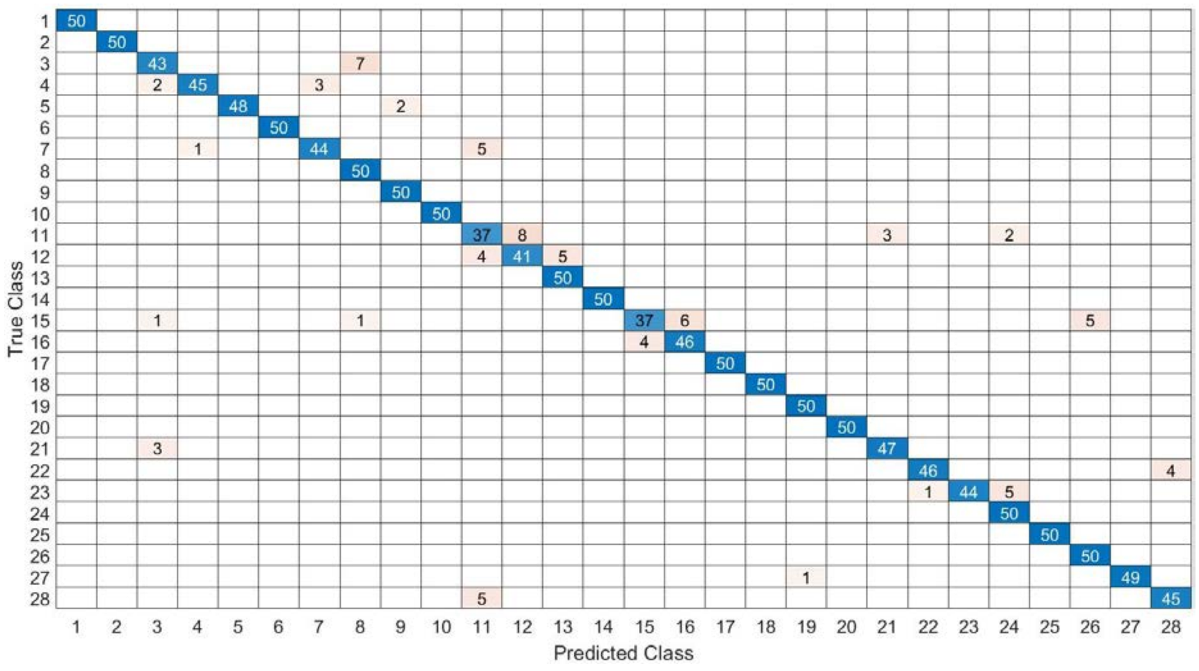
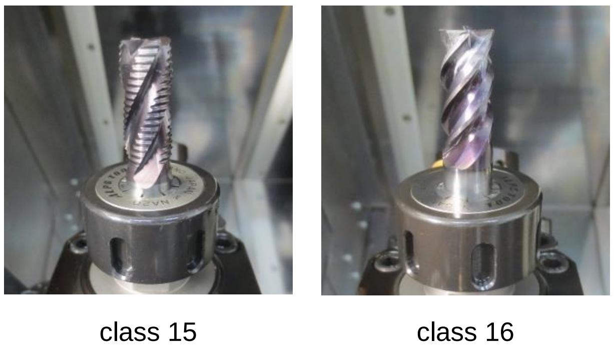
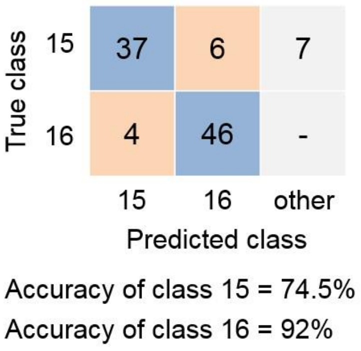


Figure 4.6 Confusion matrix of tool identification



(a) Example of two similar tools



True class \ Predicted class	15	16	other
15	37	6	7
16	4	46	-

Accuracy of class 15 = 74.5%
Accuracy of class 16 = 92%

(b) Confusion matrix of the two classes

Figure 4.7 Tool type misidentification

Furthermore, the error in each tool identification was analyzed. In tool identification, the misidentification usually occurs in tools with similar shape and dimension. Figure 4.6 depicts the confusion matrix of the tool identification results using one of the proposed prediction models (model 5). Although most predictions were correct, there were some instances of misidentification due to similar shapes/dimensions, namely between tool classes 15 and 16 (T0404 and T0505 of upper right turret), as shown in Fig. 4.7. These two tools had similar shape, color (materials), overhang, and same diameter (12 mm). Therefore, the detected features of both the tools were also almost same, and hence caused misidentification. However, the tool misidentification was less severe in deeper networks since more detail and complex features could be learned and identified by using more convolutional layers.

4.1.2.2 FDDM Predictions

The experiment of combined CNN-FDDM system used the result of VGG-19 with the decay strategy for consideration since it achieved the highest accuracy among the CNN prediction models. The results of the combined CNN and FDDM are shown in Table 4.3 and Table 4.4 for single threshold strategy and double threshold strategies, respectively. The best predictions of each strategy are marked in bold. In this stage, combinations of two detectors-descriptors were preferred, as this resulted in better inliers for the matching process. The combination of the corner and the blob-like detectors and descriptors yielded the desirable outputs. The results indicate that the use of double threshold slightly increased the accuracy compared to a single threshold.

The highest accuracy using the single threshold was 0.9850, which was obtained with matching points threshold set at 5 using the Shi Tomasi-SURF pair. The highest accuracy obtained using the double threshold was 0.986, with inliers-*MP* threshold set at 2, and matching points threshold set at 5.

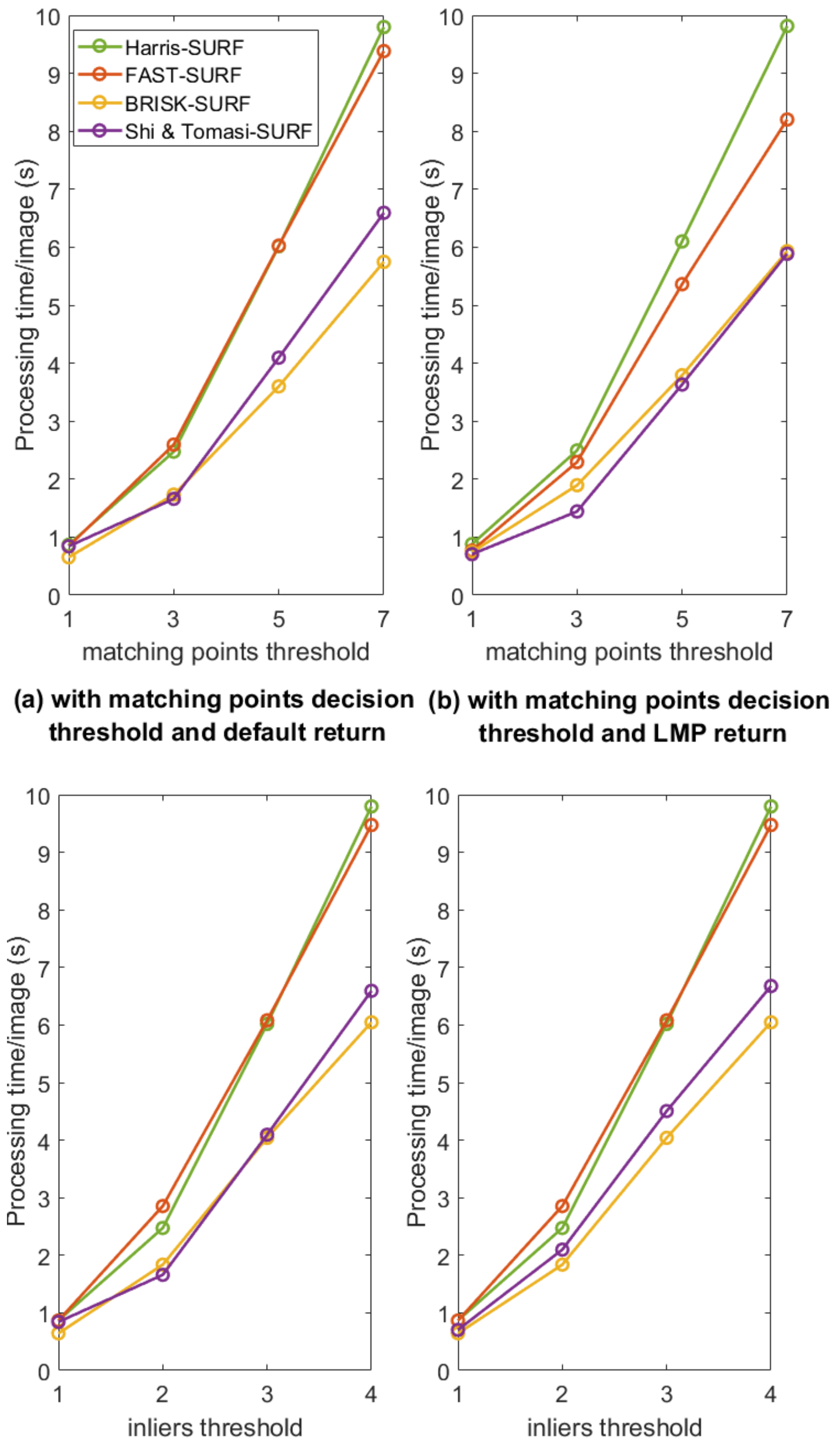
Table 4.3 Results of combined CNN-FDDM with single threshold

Detectors pair	**Inliers threshold**			**Matching points threshold**		
	Threshold λ	**Default**	**LMP**	**Threshold λ**	**Default**	**LMP**
Shi Tomasi-SURF	1	0.974	0.974	1	0.974	0.974
	2	0.976	0.976	3	0.976	0.976
	3	0.984	0.981	***5***	***0.985***	***0.983***
	4	0.983	0.979	7	0.983	0.982
Harris-SURF	1	0.974	0.974	1	0.974	0.974
	2	0.979	0.979	3	0.979	0.979
	3	0.981	0.979	5	0.978	0.976
	4	0.977	0.974	7	0.979	0.975
Fast-SURF	1	0.974	0.974	1	0.974	0.974
	2	0.981	0.979	3	0.979	0.978
	3	0.980	0.977	5	0.979	0.976
	4	0.974	0.974	7	0.979	0.976
Brisk-SURF	1	0.974	0.974	1	0.974	0.974
	2	0.979	0.978	3	0.977	0.976
	3	0.976	0.974	5	0.974	0.972
	4	0.970	0.962	7	0.976	0.969

Table 4.4 Results of combined CNN-FDDM with double threshold

Detectors pair	***MP*-inliers threshold**			**Inliers-*MP* threshold**		
	Threshold λ	**Default**	**LMP**	**Threshold λ**	**Default**	**LMP**
Shi Tomasi-SURF	1	0.976	0.976	1	0.976	0.976
	2	0.979	0.979	3	0.979	0.979
	3	0.984	0.979	***5***	***0.986***	***0.982***
	4	0.983	0.979	7	0.983	0.982
Harris-SURF	1	0.979	0.979	1	0.979	0.979
	2	0.984	0.982	3	0.984	0.981
	3	0.981	0.979	5	0.977	0.976
	4	0.978	0.975	7	0.979	0.975
Fast-SURF	1	0.979	0.978	1	0.982	0.979
	2	0.984	0.980	3	0.984	0.980
	3	0.980	0.976	5	0.976	0.976
	4	0.974	0.974	7	0.979	0.976
Brisk-SURF	1	0.977	0.976	1	0.979	0.978
	2	0.981	0.979	3	0.981	0.979
	3	0.978	0.974	5	0.974	0.971
	4	0.970	0.961	7	0.977	0.969

The addition of FDDM improved the accuracy up to 1.13 % compared to using CNN only. The decision thresholds, both based on the number of inliers and the matching points, have a similar characteristic as the hyperparameters of CNN. As in the case of the distance threshold, low decision threshold may have resulted in false predictions. Conversely, high decision threshold caused high rejection rates of the true predictions. The most optimized decision thresholds often differed, depending on the detectors-descriptors pair used and the test data.



(a) with matching points decision threshold and default return **(b) with matching points decision threshold and LMP return**

(c) with inliers decision threshold and default return **(d) with inliers decision threshold and LMP return**

Figure 4.8 Prediction time per image of combined CNN and FDDM processes

The FDDM processes were conducted using Top-5 predictions of the CNN in a sequence. That is, the comparison was started against the highest probability candidate picked by the CNN. Once the matching process yielded a result within the threshold, the process was stopped; otherwise, it continued against a lower possibility candidate. Thus, the threshold strongly influenced the processing time. In other words, as the threshold increased, processing time also increased. The processing times of combined CNN and the four approaches of FDDM are depicted in Fig. 4.8. Overall, the prediction time of the combined CNN and FDDM model was acceptable by industry standards.

Table 4.5 Results of the dimension estimation and measurement

Tool ID	**Overhang [mm]**		**Difference**		**Diameter [mm]**		**Difference**	
	est.[1]	**meas.**[2]	**abs.**[3]	**pct.**[4]	**est.**[1]	**meas.**[2]	**abs.**[3]	**pct.**[4]
Upper left turret								
T0101	14.16	13.96	0.21	1.47%		-		
T0202	40.14	39.05	1.09	2.79%		-		
T0404	36.03	34.77	1.26	3.63%	12	10.45	1.55	14.85%
T0505	40.52	37.23	3.28	8.81%	12	11.96	-	-
T0606	25.08	24.99	0.09	0.36%	6	5.57	0.43	7.74%
T0707	46.38	46.66	0.27	0.58%	5	4.39	0.61	13.95%
T0808	23.13	22.33	0.81	3.62%	8	7.22	0.78	10.86%
T0909	33.49	32.65	0.84	2.57%	-	-	-	-
T1212	62.89	62.24	0.65	1.04%	10	10.00	0.00	0.02%
Upper right turret								
T0101	33.59	31.22	2.38	7.61%	-	-	-	-
T0202	36.72	39.93	3.21	8.05%	-	-	-	-
T0303	47.41	47.04	0.37	0.78%	-	-	-	-
T0404	18.17	17.96	0.21	1.16%	8	7.91	0.09	1.16%
T0606	34.69	31.67	3.02	9.53%	12	11.00	1.01	9.14%
T0808	26.54	25.51	1.04	4.06%	8	7.66	0.34	4.38%
T0909	23.84	24.52	0.69	2.79%	6	5.90	0.10	1.64%
Lower turret								
T0101	45.43	41.36	4.07	9.84%	-	-	-	-
T0202	16.65	17.54	0.89	5.07%	8	8.70	0.70	8.06%
T0303	32.79	31.97	0.82	2.58%	6	5.69	0.31	5.41%
T0404	75.57	76.02	0.45	0.59%	6	5.82	0.18	3.02%
T0505	23.11	23.74	0.63	2.67%	8	8.38	0.38	4.50%
T0606	63.16	62.84	0.32	0.51%	10	9.60	0.40	4.20%
T0707	39.46	37.00	2.46	6.65%	12	12.05	0.05	0.37%
T0808	51.86	49.68	2.18	4.38%	-	-	-	-
T0909	54.73	51.80	2.93	5.66%	-	-	-	-
T1010	46.24	48.55	2.31	4.76%	3	3.66	0.26	7.05%
T1111	88.25	84.72	3.54	4.17%	11	12.73	2.23	17.52%
T1212	42.36	42.15	0.20	0.48%	-	-	-	-

[1]est. : estimated dimensions [2]meas. : measured dimensions
[3]abs. : absolute difference [4]pct. : percentage difference

4.1.3 Dimension Estimation and Measurement

The measurement experiment used the Keyence GT2-P12 pencil-shaped displacement sensor, equipped with a GT2-71MCN analog output amplifier unit. The communication unit DL-RS1A with an RS-232 port was used to connect the sensor set with the computer. The measurement was performed in all the installed tools. The tool path for the measurement process was automatically generated by considering the predicted type and the estimated dimensions. Table 4.5 shows the tool overhang and the diameter, estimated using the edge detection and the measured dimension obtained using a contact-based displacement sensor.

The results indicate that the estimation method yielded an acceptable prediction for overhang with a maximum difference of 9.48 % and an average difference of 3.79 % as compared to the measured dimension. The accuracy of measurement on diameter was lower with 6.70 % average difference, since only one displacement sensor was used. Ideally, the diameter measurement utilized two sensors so that both the side edges could contact the sensor.

Table 4.6 shows the processing time. The values of the image loading time were relatively similar, with an average of 0.52 s/image, as image size was uniform in the experiments. The identification process using the combined CNN and FDDM required an average of 2.22 s/image, similar to the average prediction time for the test data set. Prediction time was heavily influenced by the image quality, as unclear and blurred images lengthened the identification process.

Table 4.6 Prediction and measurement time

Tool	**Processing time [s]**				**Tool**	**Processing time [s]**			
ID	**data.**[1]	**ident.**[2]	**height**[3]	**dia.**[4]	**ID**	**data.**[1]	**ident.**[2]	**height**[3]	**dia.**[4]
Upper left turret					***lower turret***				
T0101	0.47	2.45	20.89	-	T0101	0.48	0.62	31.91	-
T0202	0.51	2.57	47.62	-	T0202	0.52	1.04	57.52	59.27
T0404	0.64	2.22	91.76	12.93	T0303	0.50	1.35	39.11	23.57
T0505	0.72	1.83	32.44	14.68	T0404	0.48	9.61	49.95	31.13
T0606	0.49	1.14	28.38	68	T0505	0.46	0.92	25.93	13.06
T0707	0.52	2.09	31.68	13.56	T0606	0.49	1.77	34.68	67.26
T0808	0.48	1.37	25.27	23.58	T0707	0.68	0.68	22.37	14.04
T0909	0.49	1.17	54.67	-	T0808	0.50	1.57	33.53	-
T1212	0.48	1.54	41.57	12.05	T0909	0.45	0.70	54.48	-
Upper right turret					T1010	0.46	1.98	61.95	13.46
T0101	0.51	1.33	31.29	-	T1111	0.71	4.75	52.75	12.93
T0202	0.49	1.07	38.53	-	T1212	0.48	0.70	45.93	-
T0303	0.53	1.39	54.97	-					
T0404	0.43	2.44	26.74	30.96					
T0606	0.46	1.58	57.73	27.95					
T0808	0.76	5.52	30.27	14.39					
T0909	0.47	6.70	24.17	31.82					

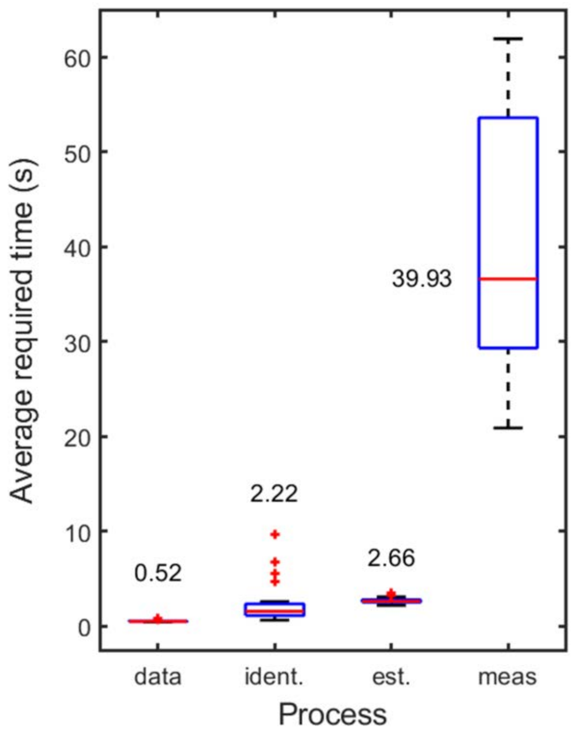
[1]data. : image loading time [2]ident. : type identification/prediction time
[3]height : height/length measurement time [4]dia. : diameter measurement time

4.1.4 System Performance Analysis

The estimation process was greatly affected by image quality and the camera setting. A tool image that was perfectly perpendicular to the camera was preferred as it left no capture angle

α, and hence, a more accurate approximation could be obtained. Another important aspect was the noise due to the workpiece scrap and machining liquid present on the tool, which could slightly alter the detected edge. Meanwhile, the error of measurement process mainly stemmed from the sampling time, tool movement speed, and linearity between the tool and the sensor. The error owing to the sensor accuracy was negligible compared to the above factors.

An important benefit of the vision-based tool recognition and dimension estimation is that it can provide a quick prediction. The overall process of recognition and estimation required an average of 5 s to yield the result, as indicated in Table 6. Therefore, it is a useful and practical method for a quick review of the installed tool. Moreover, the results also show that the measurement time was much larger than the identification time. The measurement time was influenced by the approximate dimensions and the tool movement speed, with an average length measurement time of 41.01 s/tool and a diameter measurement time of 26.92 s/tool. Figure 4.9 shows the boxplot of the average prediction and measurement times.



data	= image loading time
ident.	= tool identification time
est.	= dimension estimation time
meas.	= measurement time

Figure 4.9 The average prediction and measurement time

The long measurement time was caused due to the slow movement of the tool (feed rate in the NC program). This speed was set low considering the safety aspect since a pencil-shaped displacement sensor was used. However, it was necessary to know the length of tool accurately not only for avoiding collisions, but also because it affected the cutting performance. Furthermore, it is necessary for analyzing the effect of the tool overhang on the vibration. To summarize, the current proposed vision-method on the whole still had a considerable error (more than 0.5 mm). Therefore, the measurement process using a sensor is still the better option. Improvement can be achieved by several factors as follows:

- using other types of sensors, for example an optical sensor, so that the movement speed can be increased.
- reducing the initial distance between the tool and the sensor (less distance traveled by the tool).
- improving the vision-system by adding the image processing to sharpen the difference between the foreground and the background image, adjusting the edge detection method, developing more accurate estimation algorithms, and using a higher resolution camera.

4.2 Surface Quality Inspection

The target of the second study is to correctly predict the surface roughness and the existence of unstable cutting marks on the texture of the machined surface. The surface roughness prediction is a regression problem where the target is the numerical value of *Ra*. Since in the regression problem, the loss function suitability is strongly influenced by the training data distribution, the compatibility of the five regression loss (MSE, MAE, MAPE, Log-cosh, and Huber) with the surface roughness data was investigated.

In this section, two schemes are proposed. In the first scheme the evaluation models for roughness and unstable cutting are developed individually. The second scheme involves the development of a combined model where the training process of unstable cutting and roughness evaluation is conducted simultaneously in one model.

4.2.1 Experiment Setting

A set of machining experiments was performed to test the proposed prediction system. The workpiece materials used in the cutting process were Japanese Industrial Standards (JIS) S45C carbon steel and A5052 aluminum alloy. For the turning experiment, two types of inserts were used, Mitsubishi carbide TNGG160402R with a nose radius of 0.2 mm and TNGG160404R with a nose radius of 0.4 mm. The turning experiments were conducted on two machines, the Howa Strong 860 manual lathe and the Mazak Integrex I-100 turning center. Likewise, the experiments for milling were also conducted on two machines, the Hitachi 2MW-V manual milling machine and the Makino Seiki MSA30. Several cutting tools were used, such as the Sanko cobalt high speed steel end mill with a diameter of 10 mm, and the NS MSE230 carbide end mill-TiAlN coating with a diameter of 8 mm and 10 mm.

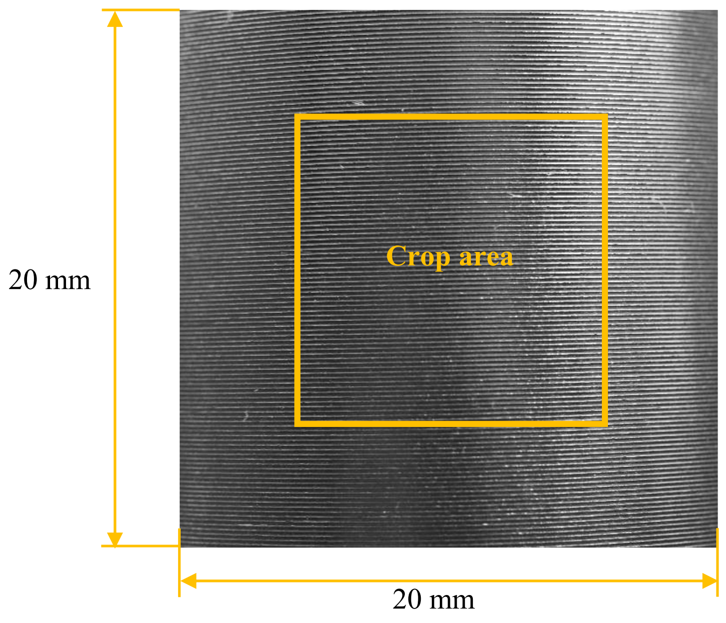
To determine the cutting parameters for the experimental procedures, the data for the cutting tools and the recommendations from the manufacturers were taken into consideration. A total of 333 machining experiments were performed on four machines by varying the cutting parameters (*S*, *C*, *F*, and *D*) to generate different roughness values. The selected cutting parameters for the experiments are given in Table 4.7. By controlling the cutting parameters, surfaces with a varying range of roughness were produced. Table 4.8 summarizes the machining experiment results for each process and material. The details of the machining experiments of the turning, slot, and side milling processes and the cutting parameters are available in Appendix A.

Table 4.7 Selected cutting parameters

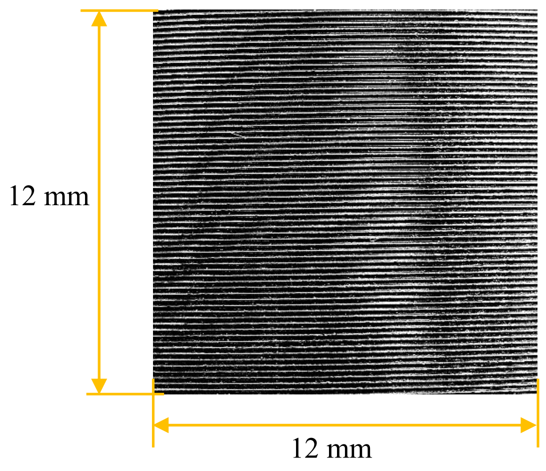
Process	**Machines**	**Parameter**	**Parameter values range**		**Specimen ID**
			Carbon steel S45C	**Aluminum alloy A5052**	
Turning	Howa Strong 860	*D* [mm]	0.2 – 1.4	–	T98S-T115S
		S [rpm]	380 – 1500	–	
		F [mm/rev]	0.10 – 0.25	–	
	Mazak Integrex I-100	*D* [mm]	0.2 – 1.4	0.2 – 1.4	T1S-T97A
		C [m/min]	100 – 245	150 – 400	
		F [mm/rev]	0.10 – 0.25	0.10 – 0.25	
Slot milling	Hitachi 2MW-V	*D* [mm]	0.5 – 2.0	0.5 – 2.0	L1S-L41A
		S [rpm]	455 – 1500	455 – 1500	
		F [mm/min]	50 – 200	15 – 200	
	Makino Seiki MSA30	*D* [mm]	0.5 – 2.0	0.5 – 2.0	L42S-L105A
		S [rpm]	2500 – 8000	1500 – 5500	
		F [mm/ min]	200 – 550	200 – 550	
Side milling	Hitachi 2MW-V	*D* [mm]	0.4 – 1.0	0.4 – 1.0	E1S-E33A
		S [rpm]	610 – 1500	610 – 1500	
		F [mm/ min]	50 – 200	100 – 350	
	Makino Seiki MSA30	*D* [mm]	0.4 – 1.0	0.4 – 1.0	E34S-E113A
		S [rpm]	1500 – 9375	1500 – 4500	
		F [mm/ min]	200 – 750	200 – 750	

Table 4.8 Machining experiments results

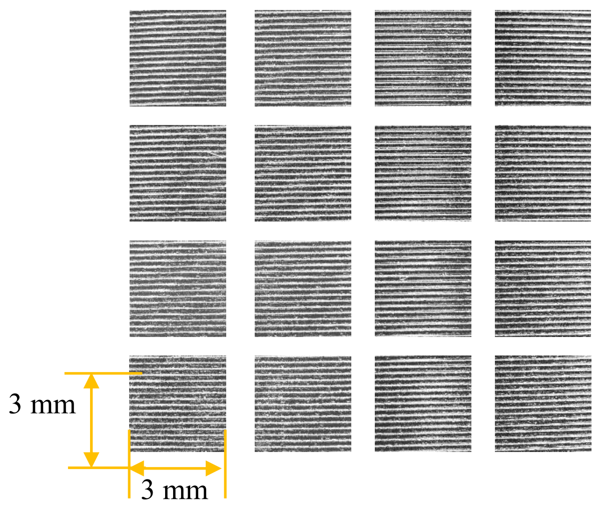
Process	**Material**	**Number of samples**	**Roughness ranges [µm]**		**Number of unstable cutting samples**
			Ra	***Rz***	
Turning	S45C	50	1.26 – 13.97	7.48 – 56.42	3
	A5052	65	1.39 – 34.06	6.61 – 159.49	13
Slot milling	S45C	51	0.45 – 8.05	2.95 – 38.53	21
	A5052	54	0.26 – 2.09	2.21 – 13.32	9
Side milling	S45C	66	0.40 – 3.42	2.42 – 14.67	29
	A5052	47	0.33 – 2.50	2.12 – 11.92	12



(a) Original captured image

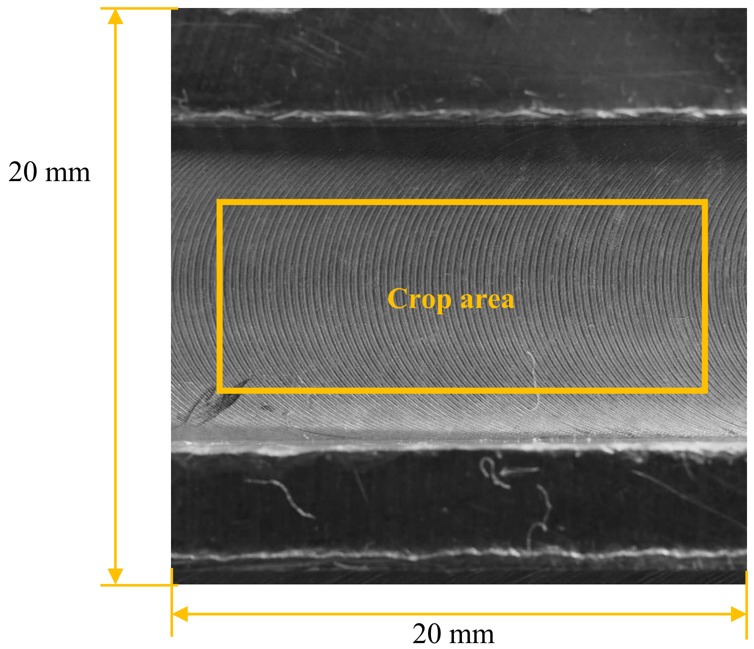


(b) Cropped image

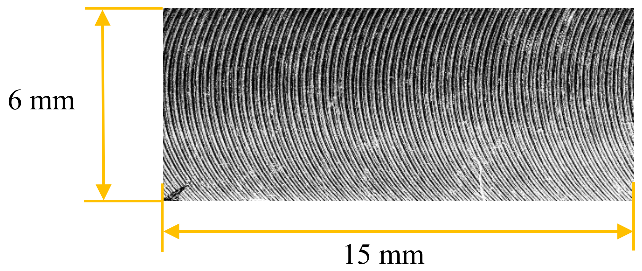


(c) Divided images

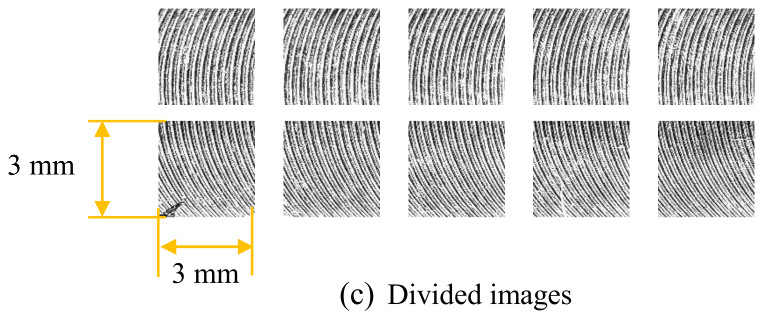
Figure 4.10 Cropping and splitting processes of turned surface image



(a) Original captured image



(b) Cropped image



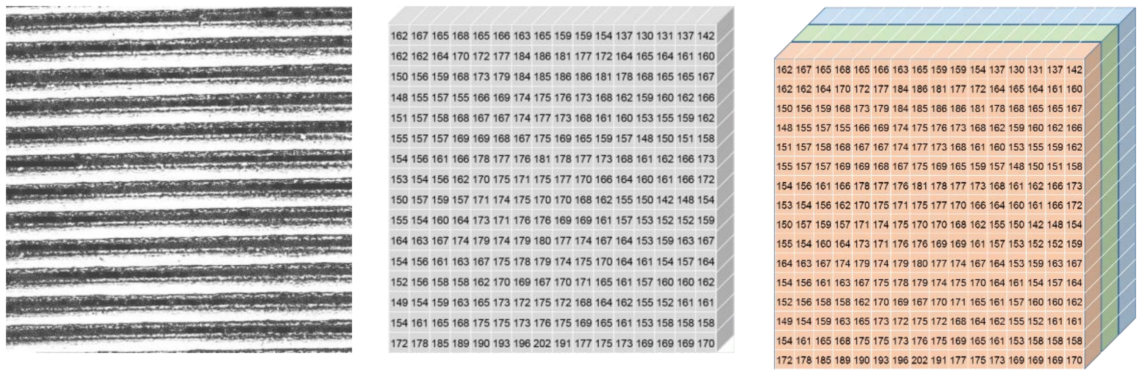
(c) Divided images

Figure 4.11 Cropping and splitting processes of milled surface image

A digital camera (Olympus OM-D E-M10 Mark II) with a macro lens (M. Zuiko Digital ED with a focal range 30 mm and a maximum magnification of 1.25x) was used to capture the image. The image acquisition used only room light without special lighting or illumination setup.

Ten images were taken for each sample, corresponding to different areas of the surface. All images were taken with a specific distance between the camera and the objects and at the same magnification because the data for both the training and the testing had to be on a uniform scale. Figures 4.10 and 4.11 show the cropping process and the scale of individual images for turned and milled surface data, respectively.

Due to the camera setting (i.e., focus, aperture, shutter speed) and lighting setting, not all areas of the image were usable. Therefore, the image was center cropped since the central area had the clearest feed marks pattern. Next, the cropped image was divided into 16 sub-images for turned surface, and 10 sub-images for the milled surface. Each sub-image represented 3×3 mm area of the machined surface.

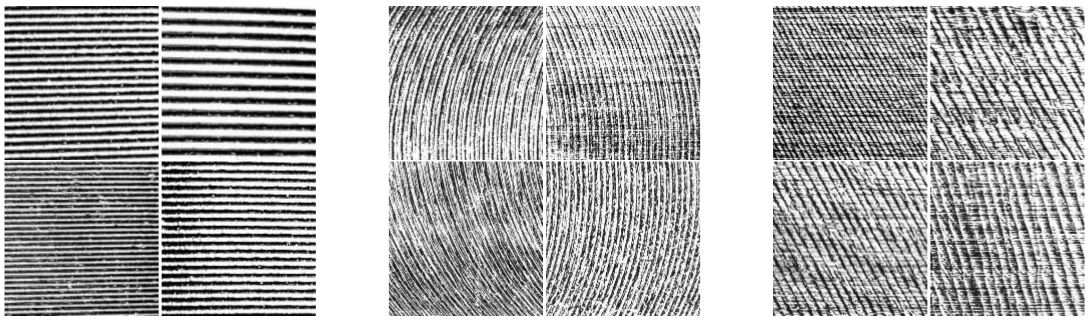


(a) Sample image

(b) Grayscale data
Data size: 500×500

(c) RGB data
Data size: 500×500×3

Figure 4.12 Comparison of grayscale and RGB data format



(a) Turned surface

(b) Slot milled surface

(c) Side milled surface

Figure 4.13 Image of samples after pre-processing

The individual input data of the deep learning model was the divided image. The image was in grayscale format in which each pixel represented an amount of light that only carried the intensity information. The deep learning model building of this study used a single 256 pixel-depth with a value range of 0 – 255 that carried the intensities of light (shades of gray). Since the surface quality inspection problem emphasizes the identification of ridge and valley patterns created by the cutting process, the grayscale format was adequate to highlight the differences between those two regions. Moreover, it also drastically reduced the processing time compared to other formats such as RGB which had a 3D matrix format for each image. Figure 4.12 shows the comparison of the data format of a 500×500-pixel sample image in grayscale and RGB.

Next, data augmentation and image processing were performed both to enhance the features and to multiply the images. The prediction models for each machining process were developed independently because each process had distinct patterns of ridges and valleys. Figure 4.13 shows the image samples of three machining processes after pre-processing.

The surface profile of the workpieces was then measured using the contact-based profilometer Surfcom Flex 50A. The measurement of each sample was carried out three times, and then averaged to determine the arithmetic mean surface roughness *Ra*. The measured *Ra* and *Rz* values of each sample are listed in appendix A. The measurement conditions are listed in Table 4.9.

Table 4.9 Measurement conditions

Standard	JIS-2001
Evaluation length	24 mm
Cut-off	0.8 mm
Cut-off filter	Gaussian
Tip radius	2.5 µm
Drive speed	0.6 mm/s
Measuring force	0.75 mN
Resolution	0.001 µm

The prediction experiments were conducted in two schemes. In the first scheme, cross test was used where the test sets were drawn from the training set. The test sets contained different images but within the same sample with the training sets, thus having the same target value. In this case, 10% images of each sample were randomly selected as the test sets. This scheme later is referred as *cross-test* on the performance evaluation.

Table 4.10 Number of images after pre-processing

Datasets	**Cross-test**			**Cases 1-3**		
	Training	**Validation**	**Test**	**Training**	**Validation**	**Test**
Turning	58,880	7,360	7,360	60,480	6,720	6,400
Slot milling	33,280	4,200	4,200	34,200	3,800	4,000
Side milling	36,160	4,520	4,520	37,080	4,120	4,000

In the second scheme, the test sets belonged to different samples from the training and the validation sets. Therefore, the test sets were independent of the training data and have never been learned by the trained model. For each machining data, 10 samples were randomly selected for test sets. This process was repeated three times, so that in this scheme there were three cases for each machining data. The number of images in each scheme after pre-processing for training, validation, and test are described in Table 4.10. In addition, Table 4.11 shows the test datasets and the properties for each surface sample. The sample ID represent the material of the specimen, i.e. T111S uses S45C carbon steel, while T35A uses A5052 aluminum alloy.

Table 4.11 Test data sets

Turned surfaces											
Case 1	Sample ID	T111S	T115S	T2S	T20S	T30S	T35A	T37A	T40A	T63A	T88A
	Ra [μm]	2.42	3.88	11.21	10.58	8.77	7.24	11.75	11.60	22.35	6.19
	Unstable	No	Yes	No	No	No	No	Yes	No	Yes	No
Case 2	Sample ID	T114S	T5S	T6S	T24S	T36A	T42A	T44A	T62A	T74A	T90A
	Ra [μm]	4.06	8.45	11.30	12.14	11.44	4.98	11.51	24.39	4.20	8.26
	Unstable	Yes	No	No	No	No	No	No	Yes	No	Yes
Case 3	Sample ID	T99S	T16S	T27S	T28S	T33A	T46A	T48A	T61A	T71A	T94A
	Ra [μm]	2.26	5.16	9.04	12.71	5.58	5.28	11.93	17.06	4.04	11.92
	Unstable	No	No	No	No	Yes	No	No	Yes	No	Yes
Slot milled surfaces											
Case 1	Sample ID	L7S	L16S	L48S	L56S	L70S	L82A	L85A	L22A	L101A	L41A
	Ra [μm]	5.77	4.50	0.56	0.82	0.77	0.52	0.74	1.22	0.66	0.65
	Unstable	No	No	No	Yes	Yes	No	Yes	No	No	No
Case 2	Sample ID	L5S	L11S	L43S	L57S	L75S	L78A	L89A	L29A	L104A	L35A
	Ra [μm]	3.20	4.66	1.11	0.85	0.92	0.51	0.80	0.70	0.71	0.76
	Unstable	No	No	Yes	Yes	Yes	No	Yes	No	Yes	No
Case 3	Sample ID	L6S	L42S	L50S	L62S	L71S	L77A	L80A	L26A	L91A	L38A
	Ra [μm]	4.85	0.86	1.33	0.80	0.71	0.43	0.85	1.09	0.62	0.68
	Unstable	No	Yes	Yes	No	Yes	No	Yes	No	No	No
Side milled surfaces											
Case 1	Sample ID	E5S	E14S	E34S	E83A	E91A	E62S	E19A	E28A	E79S	E106A
	Ra [μm]	1.15	1.70	0.68	0.41	0.58	0.56	0.36	0.36	1.83	1.43
	Unstable	No	Yes	No	No	Yes	Yes	No	No	Yes	Yes
Case 2	Sample ID	E9S	E17S	E48S	E82A	E87A	E55S	E24A	E32A	E77S	E110A
	Ra [μm]	1.47	1.81	0.57	0.43	0.44	0.63	0.38	0.38	0.59	1.42
	Unstable	Yes	No	Yes	No	No	Yes	No	No	No	Yes
Case 3	Sample ID	E18S	E35S	E43S	E88A	E90A	E58S	E23A	E27A	E70S	E104A
	Ra [μm]	1.61	0.64	0.64	0.41	0.62	0.53	0.33	0.35	1.07	0.55
	Unstable	No	No	No	No	Yes	Yes	No	No	Yes	No

4.2.2 Performance Evaluation Measures

The adequacy of the prediction by the developed models was judged based on the type of problem. As the unstable cutting mark identification problem is a binary problem, the performance was simply measured by its error rate as formulated in Eq. (4.3).

$$E_{ch} = \frac{1}{n}\sum_{x} q_x \tag{4.3}$$

where E_{ch} is error rate of the unstable cutting mark identification and n is the number of samples. q_x is the binary variable of the unstable cutting mark prediction that will have a value of 1 if it is a correct prediction, and 0 otherwise. The accuracy of the model A_{ch} can be calculated as A_{ch} = 1- E_{ch}. Since the unstable cutting mark identification is a binary classification task, only Top-1 accuracy γ_1 was evaluated.

The performance of surface roughness estimation was evaluated using several indices: mean absolute percentage error (MAPE$_{Ra}$), root mean square error (RMSE), coefficient of determination R^2, and adjusted-R^2. MAPE$_{Ra}$ is a straightforward measure which shows the error rate of the models. It follows the same equation as MAPE loss described in Section 3.3.2.2. In the case of the performance measure, the equation of MAPE is described by the following formula.

$$MAPE_{Ra} = \frac{1}{n}\sum_{x}\left|\frac{p_x - q_x}{q_x}\right| \tag{4.4}$$

where p_x is the target (real) *Ra* value of the sample x measured using the profilometer and q_x is the predicted *Ra* value of the sample x obtained by the prediction model. Alternatively, the accuracy of the models A_{Ra} can be calculated as A_{Ra} = 1- MAPE$_{Ra}$. However, the accuracy often does not show the performance objectively since it is value-dependent on the denominator. Therefore, RMSE, R^2, and adjusted-R^2 are added for the model performance comparison, as formulated in Eqs. (4.4-4.6).

$$RMSE_{Ra} = \sqrt{\frac{1}{n}\sum_{x}(p_x - q_x)^2} \tag{4.4}$$

$$R^2 = 1 - \frac{\sum_{x}(p_x - q_x)^2}{\sum_{x}(p_x - \bar{p})^2} \tag{4.5}$$

$$adjusted - R^2 = 1 - \left(\frac{n-1}{n-d-1}\right)\frac{\sum_{x}(p_x - q_x)^2}{\sum_{x}(p_x - \bar{p})^2} \tag{4.6}$$

where $\bar{p}$ is the mean of the target data and d is the degree of the polynomial. MAPE is fundamentally easier to understand than RMSE since each error influences MAPE in direct proportion to the absolute value of the error, which is not the case for RMSE [110]. Furthermore, RMSE does not show the proportion of error over its target or predicted value. As such, it cannot be used to compare the prediction modes performance over two or more different datasets.

The test procedure for unstable cutting mark identification followed the same training-test data allocation as the surface roughness estimation problem. The error rate for unstable cutting marks identification E_{ch} was simply calculated as the proportion of incorrect predictions over all data, whereas the accuracy was calculated as the number of correct predictions over total number of samples.

4.2.3 Performance of the Surface Roughness Estimation Model

4.2.3.1 Effect of Architecture

Experiments were conducted using the same hyperparameter settings for all the architectures. Similar to tool type identification, learning rate schedule of gradient descent was used to allow a large search area of the loss landscape at the beginning of the iterations which was gradually reduced as convergence was approached. The learning rate schedule was overseen by the decay step ε and the decay factor c.

Table 4.12 Results of the training and prediction process for turned surfaces

Model	**Case**	**RMSE**	***R***	***R*^{2}**	**Adjusted - *R*^{2}**	**Average accuracy**	**Training time *T* [h]**	**Test time /sample τ [s]**
1	cross	0.47	1.00	0.99	0.99	93.78%	26.85	0.63
	1	1.13	0.98	0.96	0.96	90.58%	32.66	0.64
	2	1.17	0.98	0.97	0.96	91.24%	27.57	0.61
	3	2.61	0.94	0.88	0.88	86.35%	26.85	0.62
2	cross	0.49	1.00	0.99	0.99	93.53%	24.34	0.37
	1	1.28	0.97	0.94	0.94	89.95%	25.63	0.38
	2	1.18	0.98	0.97	0.97	91.24%	25.76	0.37
	3	2.40	0.95	0.90	0.90	87.44%	26.16	0.37
3	cross	0.46	1.00	0.99	0.99	94.17%	56.87	3.05
	1	1.49	0.97	0.94	0.93	89.06%	56.26	2.99
	2	1.18	0.99	0.97	0.97	93.86%	59.88	3.30
	3	2.18	0.95	0.89	0.89	85.47%	58.35	2.98
4	cross	0.30	1.00	1.00	1.00	96.12%	51.01	1.87
	1	1.28	0.98	0.95	0.93	88.18%	51.26	1.97
	2	1.18	0.99	0.97	0.97	93.26%	49.88	2.02
	3	2.21	0.94	0.89	0.89	87.07%	58.35	1.96

Unlike the fixed number of epochs used in tool type identification, validation-based early stopping was used in this study to prevent over-fitting and reduce the overall training time.

In early stopping, the training process was stopped when there was no indication of improvement on the loss of validation sets after a certain number of iterations. Therefore, the training time *T* was calculated as the total time required to learn all the training samples until the 'stop criterion' was fulfilled. On the other hand, the test time τ was defined as the average time required to predict one original image. It was calculated as the aggregate of the sub-image prediction times. Tables 4.12 – 4.14 show the results of the surface roughness prediction on turned, slot milled, and side milled surface data, respectively.

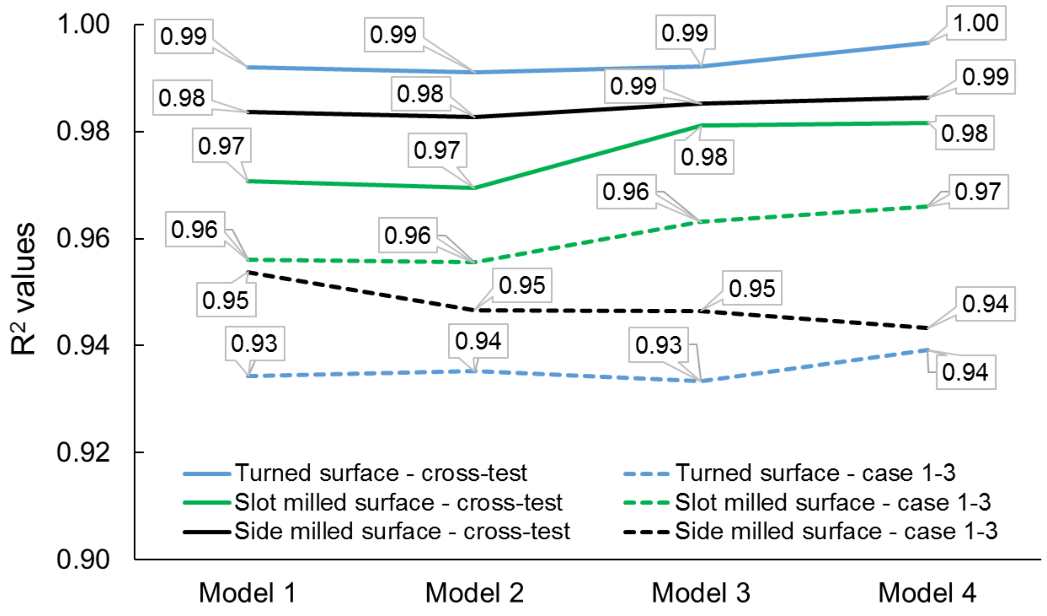
Table 4.13 Results of the training and prediction process for slot milled surfaces

Model	**Case**	**RMSE**	***R***	***R***^{2}	**Adjusted - *R***^{2}	**Average accuracy**	**Training time *T* [h]**	**Test time /sample τ [s]**
1	cross	0.27	0.99	0.97	0.97	90.34%	21.83	0.38
	1	0.35	1.00	0.99	0.99	88.62%	22.10	0.39
	2	0.48	0.95	0.91	0.91	82.61%	23.28	0.39
	3	0.22	0.99	0.97	0.97	86.55%	25.76	0.39
2	cross	0.29	0.98	0.97	0.97	87.95%	13.74	0.23
	1	0.29	1.00	0.99	0.99	88.98%	14.85	0.23
	2	0.49	0.95	0.91	0.90	81.73%	15.15	0.22
	3	0.22	0.98	0.97	0.97	87.64%	12.83	0.22
3	cross	0.22	0.99	0.98	0.98	92.64%	39.07	1.91
	1	0.30	0.99	0.99	0.99	87.80%	39.08	1.91
	2	0.53	0.96	0.92	0.92	83.95%	43.39	1.91
	3	0.17	0.99	0.98	0.98	84.85%	43.23	1.92
4	cross	0.22	0.99	0.98	0.98	93.25%	34.48	0.98
	1	0.32	1.00	0.99	0.99	90.52%	28.95	1.01
	2	0.46	0.96	0.92	0.92	84.71%	31.60	1.02
	3	0.19	0.99	0.98	0.98	86.48%	38.77	1.04

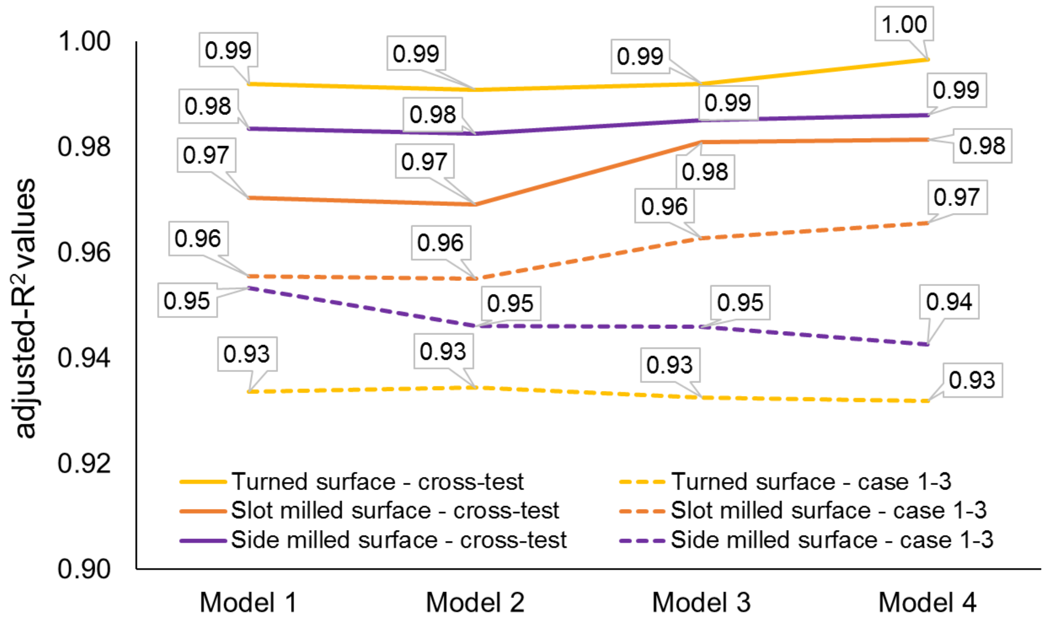
Table 4.14 Results of the training and prediction process for side milled surfaces

Model	**Case**	**RMSE**	***R***	***R***^{2}	**Adjusted - *R***^{2}	**Average accuracy**	**Training time *T* [h]**	**Test time /sample τ [s]**
1	cross	0.08	0.99	0.98	0.98	94.49%	11.49	0.38
	1	0.15	0.98	0.96	0.96	90.32%	13.51	0.38
	2	0.20	0.97	0.95	0.95	90.92%	10.44	0.38
	3	0.08	0.98	0.95	0.95	90.25%	8.82	0.38
2	cross	0.08	0.99	0.98	0.98	93.99%	7.44	0.22
	1	0.13	0.98	0.95	0.95	92.03%	10.33	0.22
	2	0.18	0.98	0.95	0.95	90.73%	9.28	0.23
	3	0.11	0.97	0.93	0.93	88.23%	8.16	0.23
3	cross	0.07	0.99	0.99	0.99	94.85%	32.16	2.00
	1	0.13	0.98	0.96	0.96	91.47%	36.56	1.93
	2	0.17	0.98	0.95	0.95	91.39%	35.58	1.95
	3	0.12	0.96	0.92	0.92	88.80%	33.77	1.95
4	cross	0.07	0.99	0.99	0.99	94.96%	12.97	1.04
	1	0.13	0.97	0.94	0.94	91.09%	15.08	1.02
	2	0.18	0.98	0.96	0.96	90.30%	12.45	1.01
	3	0.12	0.97	0.93	0.93	89.08%	16.57	1.02

The result demonstrated that the error of the milled surface data was, in general, slightly higher than that of the turned surface data, although the same architecture and configuration were used. This lower performance was owing to the relatively less clear and finer ridge-valley patterns in the milled surface, which hindered the feature identification process. In addition, the milled surface data had a smaller surface roughness range than the turned surface data.



(a) R^2 values



(b) adjusted-R^2 values

Figure 4.14 R^2 and adjusted-R^2 of prediction models

Because the models have learned the test samples during training as well, the performance in *cross-test* was better than that in the other three cases which used different samples for the testing dataset. Figure 4.14 shows the comparison of R^2 of the prediction models and adjusted-R^2 values on *cross-test* and cases 1 – 3.

The results also confirmed that a deeper network resulted in a higher accuracy. In turning data, the average accuracies of each model were 90.49 %, 90.54 %, 90.64 %, and 91.16 % for networks with architectures 1, 2, 3, and 4 respectively. In slot milling data, network architecture 4 again achieved the highest average accuracy of 88.74 %, followed by network 3 with 87.31 %, network 1 with 87.03 %, and network 2 with 86.58 %. Meanwhile, network 3 achieved the highest accuracy in side milling data with 91.63 %, followed by network 1 with 91.50 %, network 4 with 91.36 %, and network 2 with 91.25 %.

In all cases, architecture 3 and 4 appeared to give better outputs due to their larger input size and deeper network. Since a deeper network involved more parameters (weight and biases), more complex and non-linear functions could be learned. In addition, the advantage of multiple convolutional layers was that they could learn features at various levels of abstraction. However, a deeper network apparently required longer processing time both for training and testing because more parameters were engaged. The longer prediction time diminished the advantage of vision-based inspection since the models were expected to provide a quick prediction of the surface roughness. In addition, the deeper networks required large memory and faster GPU. Overall, considering accuracy, processing time, and resource usage, narrower models might be more realistically applied in the manufacturing process.

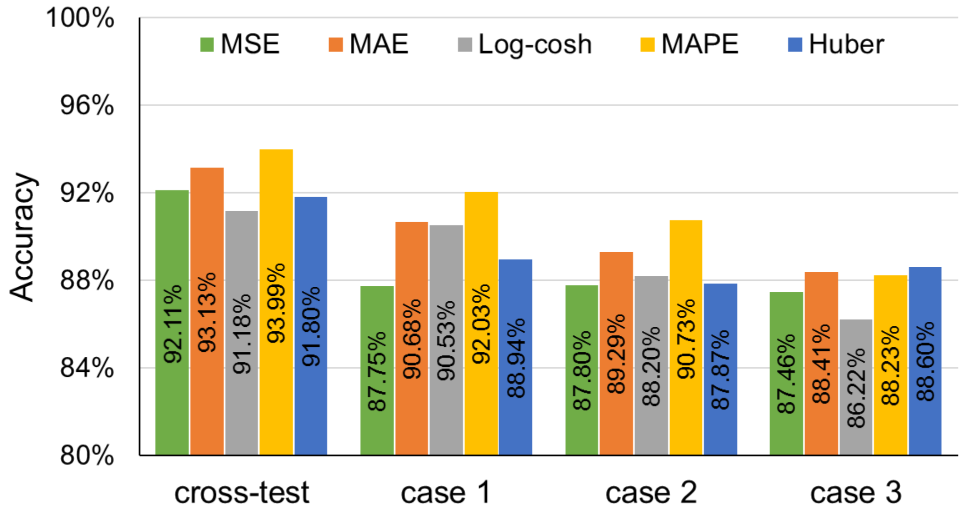
4.2.3.2 Effect of Loss Functions

To test the performance of the prediction models using various loss functions, a set of experiments was conducted using architecture 2 which had the most optimum accuracy-efficiency ratio. The tests were performed in all the cases on side milling data which had comparatively less clear patterns than the turning and the slot milling images. The training hyperparameters — learning rate lr, batch size B, maximum epoch number E, momentum m, regularization factor λ, decay step ε, and decay factor c were set to be uniform throughout the experiments. Only the hyperparameter δ for Huber loss was set according to the distribution of each data set. Table 4.15 describes the hyperparameter values used in the experiment.

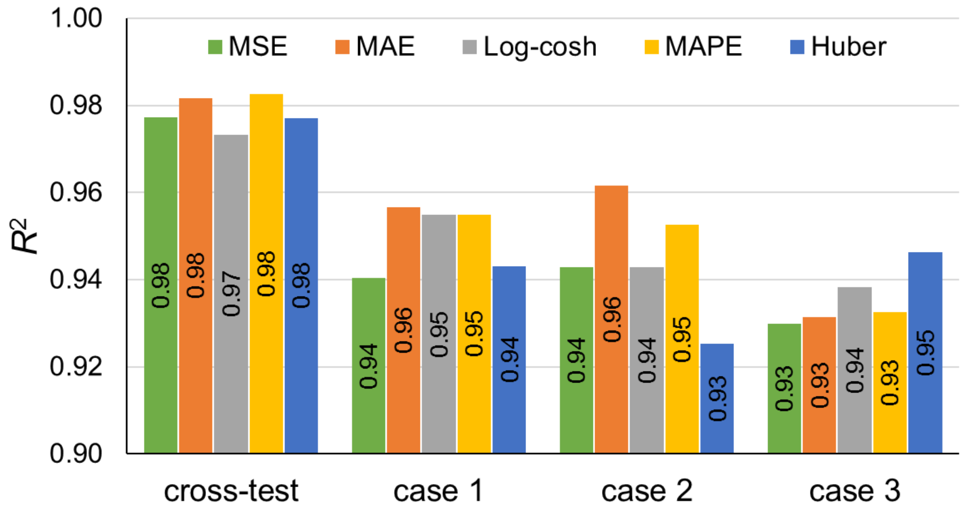
First, the experiments were performed using a separate system which built the prediction models only for the surface roughness estimation. The obtained accuracy and R^2 of the prediction models using various loss functions in side milled surface data are depicted in Fig. 4.15.

Table 4.15 Training hyperparameter value for the experiments

Hyperparameter	**Symbol**	**Value**
learning rate	*lr*	10^{-5}
batch size	*B*	10
maximum epoch	*E*	25
momentum	η	0.99
regularization factor	λ	0.001
decay step	ε	4
decay factor	*c*	0.5
Huber cut-off hyperparameter		
Turned surface	δ_T	4.0
Slot milled surface	δ_L	1.0
Side milled surface	δ_I	0.5



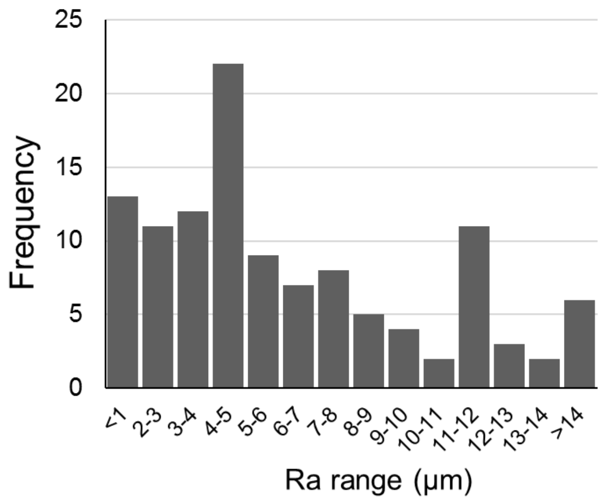
(a) Accuracy comparison



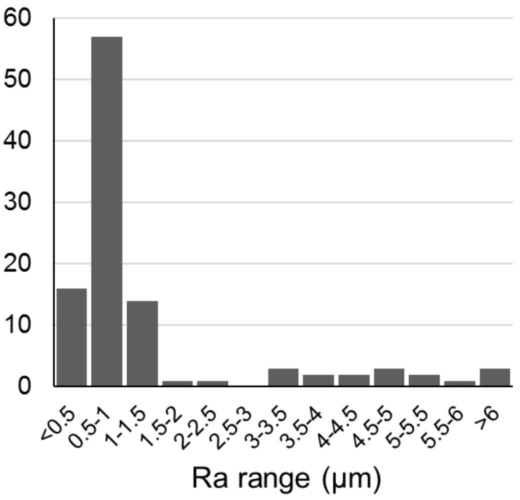
(b) R^2 comparison

Figure 4.15 Comparison of accuracy and R^2 value of the prediction models using various loss functions in side milled surface data

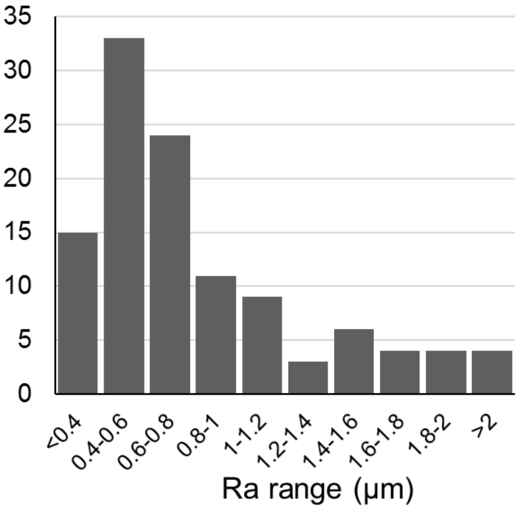
The results indicate that the four alternative loss functions outperformed the original MSE function. This was owing to the data distribution; MSE loss is suitable for the data that follow the Gaussian distribution. Meanwhile, the three data sets in surface quality inspection clearly did not follow the normal distribution. The data distributions of the data sets are depicted in Fig. 4.16.



(a) Turned surface data



(b) Slot milled surface data



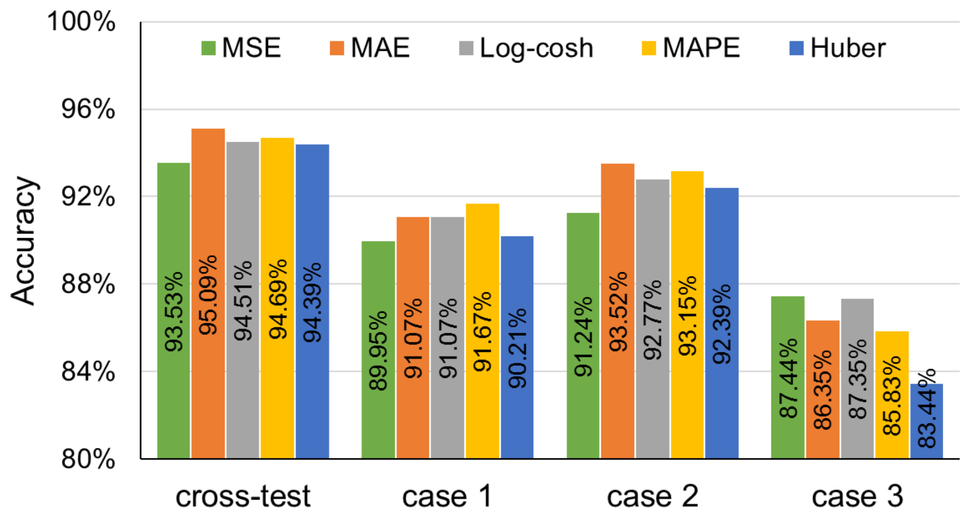
(c) Side milled surface data

Figure 4.16 Histogram of the turning, slot milling, and side milling data

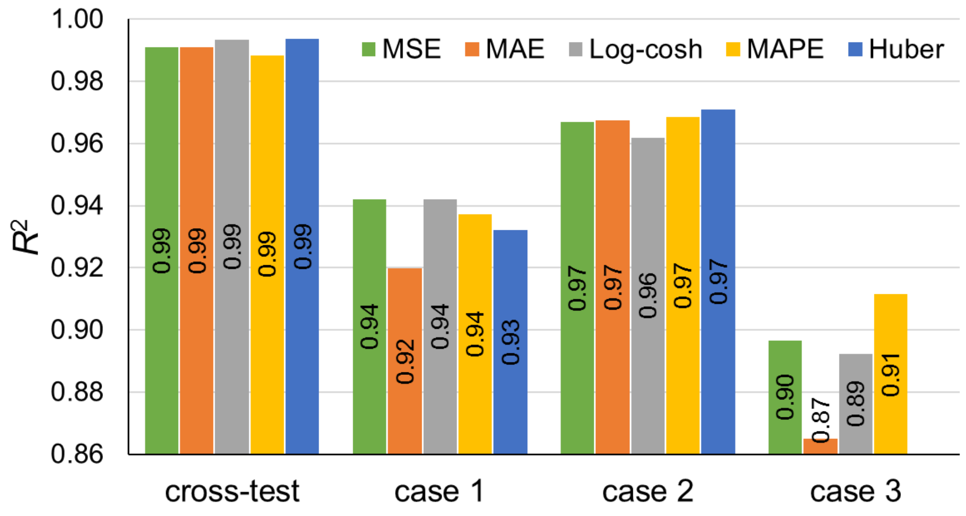
As described in section 3.3.2.2, MSE was at a disadvantage when outliers were present due to which the squaring resulted in a very high loss even though the number of outliers was small. Since MSE squares the error, the impact of the error on the higher value data (i.e. surfaces with *Ra* > 3 μm) was much bigger than the smaller value data (i.e. surfaces with *Ra*

< 1 µm). This may have skewed the metric towards overestimating these outlier errors, thus leading to lower accuracy than the other loss functions models.

The average accuracies of the prediction models in the highest to lowest order were as follows: MAPE 91.25 %, MAE 90.38 %, Huber 89.30 %, Log-cosh 89.03 %, and MSE 88.78 %. This result was then corroborated by the R^2 comparison in which MAPE models performed more favorably than the other loss functions. Furthermore, the prediction experiments using the five loss function models were conducted in turned and slot milled surface data. Figures 4.17 and 4.18 show the performance comparison between the prediction models with various loss functions in turned surface data and slot milled surface data, respectively.

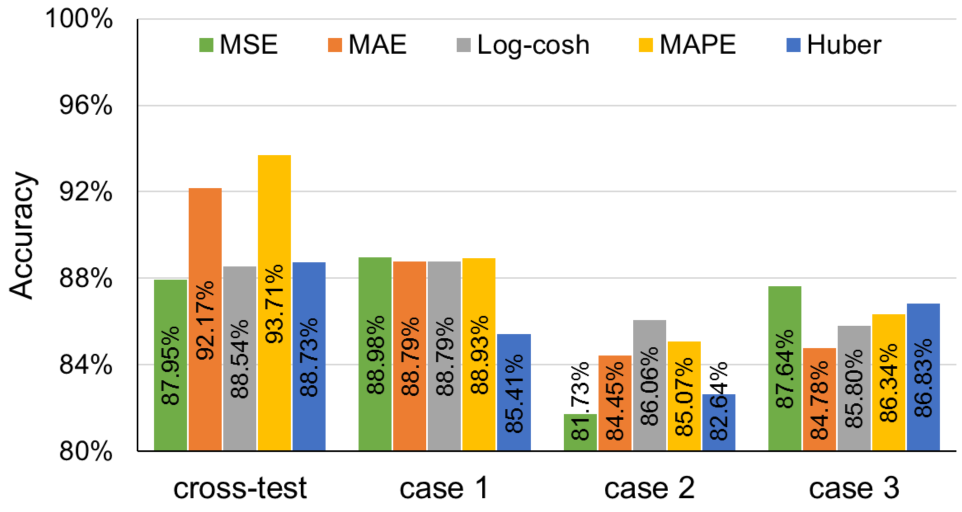


(a) Accuracy comparison

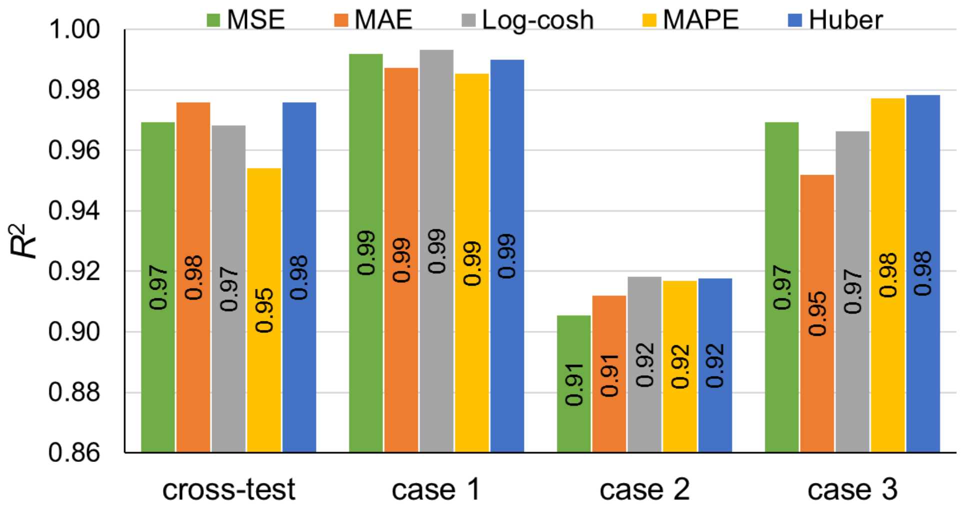


(b) R^2 comparison

Figure 4.17 Performance of prediction models in turning data



(a) Accuracy comparison



(b) R^2 comparison

Figure 4.18 Performance of prediction models in slot milling data

The results of the turned and slot milled surface data were consistent with the previous side milled data. Overall, prediction models using the MAPE loss function performed the best. On an average, the use of MAPE as a loss function improved the average accuracy by 0.79 % in turning data and 1.93 % in slot milling data compared to MSE. This is because the loss function was directly related to the evaluation measure accuracy A_{Ra}. Furthermore, in contrast to MSE, it was less susceptible toward outliers. The results also indicated that there was no significant difference in the training and prediction time between models using various loss functions. The complete results of the training and prediction processes using the five loss functions are detailed in Appendix B.

4.2.3.3 Training Speed

Figure 4.19 shows the average training and prediction (test) times of models with various loss functions. The training time was calculated as the total time required to learn all the training samples until the 'stop criterion' was fulfilled. On the other hand, the test time was defined as the average time required to predict one original image.

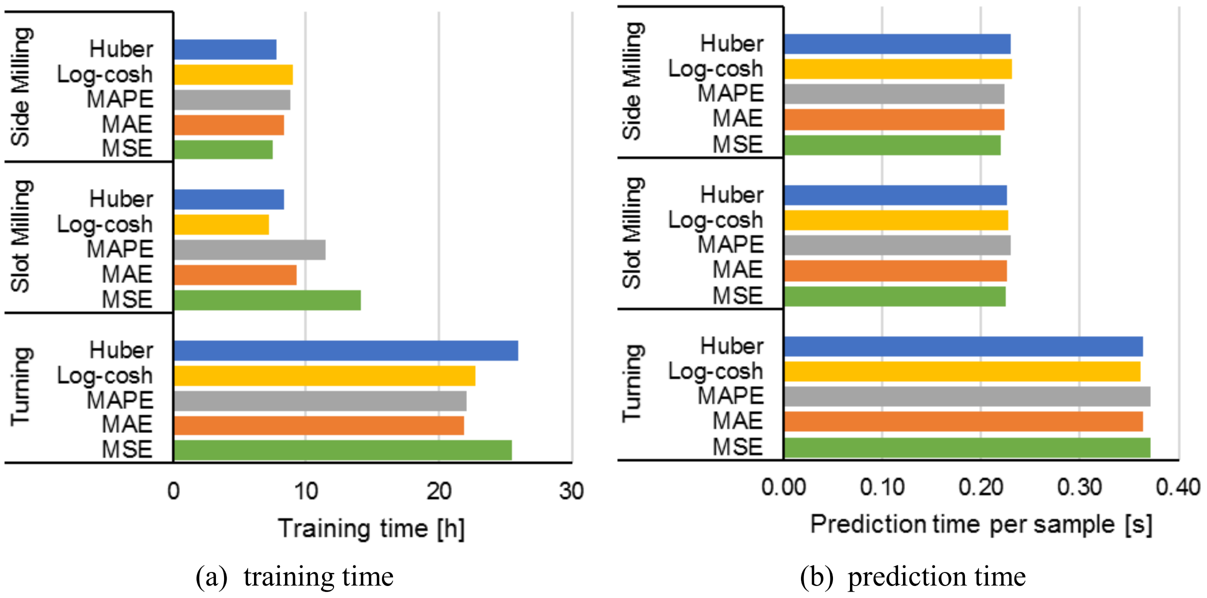


Figure 4.19 Average training time and test time of models with various loss functions

Since all the models were trained using the same architecture, theoretically there was no significant difference in the training and prediction times, sample-wise. The prediction time of the turned surface data was higher owing to a greater number of sub-images required for making a prediction (16 fragments compared to 10 fragments for a milled surface). Similarly, although the total training time appeared to be different in each model and dataset, this was actually due to different number of samples and epochs until the 'stop criterion' was fulfilled. As the training time was directly related to the convergence, it is important to analyze the training speeds because they represent how fast the learning process reached convergence. Faster training speed translated to shorter total training time since it required less epochs. Figure 4.20 shows a comparison of the speeds of the prediction models using the five loss functions in the cross-test of the slot milled data.

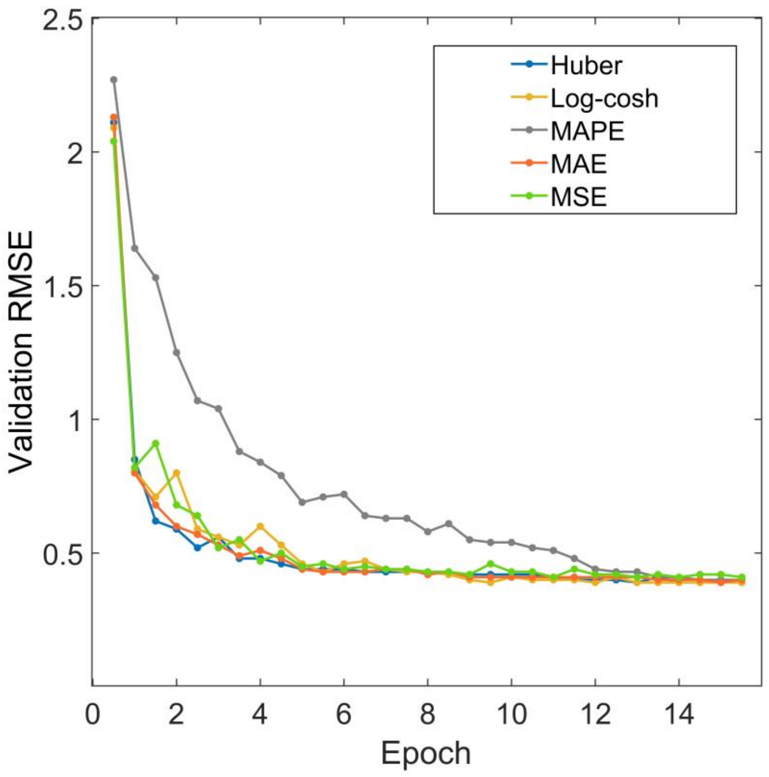


Figure 4.20 Training speed comparison

Since the loss function values differed among the models, RMSE was used as the indicator of the training quality. The faster the value decreased, the higher the training speed of the model. The validation RMSE change was monitored until epoch 15, although in some cases the training continued beyond this point. As all compared models were developed using the full training, the weight initialization was performed randomly. Therefore, the change of weights and the training speed were solely influenced by the choice of the loss function.

All the models except MAPE recorded a similar pattern of learning progress. The graph shows patterns of rapid training at an early stage. Subsequently, the models slowed down after 5 epochs at around 0.5 validation RMSE. On the other hand, the learning progress of MAPE model was slower and decreased the RMSE gradually. This may be attributed to the gradients of MAPE which caused less aggressive adjustment of the weights compared to the other loss functions. Nevertheless, all the models faced stagnation after 12 epochs before reaching the validation-based early stopping.

4.2.4 Performance of Unstable Cutting Mark Identification Model

The training process of unstable cutting mark identification models followed the same case datasets and the hyperparameters settings as those used with the surface roughness prediction models. Tables 4.16-4.18 show the results of the unstable cutting mark identification on turned, slot milled, and side milled surfaces, respectively.

Table 4.16 Results of the unstable cutting evaluation models for turned surfaces

Model	Case	Accuracy	Training time [h]	Test time /sample [s]	Model	Case	Accuracy	Training time [h]	Test time /sample [s]
1	cross	100%	20.55	0.67	3	cross	100%	51.56	2.89
	1	99%	19.05	0.65		1	99%	51.99	3.00
	2	100%	22.54	0.40		2	100%	57.63	2.98
	3	100%	21.14	0.41		3	100%	51.99	3.00
2	cross	100%	23.58	0.39	4	cross	100%	40.73	1.94
	1	99%	26.78	0.40		1	100%	41.99	1.72
	2	100%	20.94	0.39		2	100%	38.55	1.74
	3	100%	22.13	0.42		3	100%	39.25	1.81

Table 4.17 Results of the unstable cutting evaluation models for slot milled surfaces

Model	Case	Accuracy	Training time [h]	Test time /sample [s]	Model	Case	Accuracy	Training time [h]	Test time /sample [s]
1	cross	100%	15.65	0.40	3	cross	100%	25.95	1.52
	1	91%	20.21	0.41		1	87%	23.64	1.50
	2	89%	15.57	0.40		2	96%	24.64	1.51
	3	88%	10.33	0.41		3	100%	23.77	1.51
2	cross	100%	9.80	0.24	4	cross	100%	17.55	1.04
	1	97%	12.56	0.24		1	87%	17.32	1.02
	2	96%	11.60	0.23		2	82%	18.40	1.03
	3	90%	10.80	0.24		3	100%	20.30	1.03

Table 4.18 Results of unstable cutting evaluation models for side milled surfaces

Model	Case	Accuracy	Training time [h]	Test time /sample [s]	Model	Case	Accuracy	Training time [h]	Test time /sample [s]
1	cross	100%	9.90	0.40	3	cross	99%	42.53	1.95
	1	96%	12.54	0.41		1	99%	36.24	1.94
	2	66%	13.79	0.39		2	79%	37.83	2.07
	3	90%	19.07	0.39		3	93%	41.56	2.03
2	cross	97%	8.47	0.23	4	cross	98%	15.20	1.04
	1	100%	14.76	0.26		1	99%	15.17	1.06
	2	68%	7.82	0.23		2	76%	11.47	1.08
	3	86%	8.35	0.23		3	92%	11.60	1.06

Unlike the roughness estimation, the unstable cutting evaluation generally achieved very high accuracy with some models obtaining a perfect prediction. The unstable cutting evaluation problem was considerably simpler than the roughness estimation, because it was a classification problem with only two output choices, unstable cutting or stable cutting. The ease of identifying the unstable cutting marks was especially evident in the turned surface data, as the cutting marks were clearly visible. Meanwhile, the unstable cutting marks were less discernible in side milling data, which resulted in a lower identification accuracy.

In addition, the training process for the unstable cutting evaluation took less time than that for the roughness estimation. This is because the calculation process for the classification took less time than the regression used in the roughness estimation. Furthermore, the training process for the unstable cutting evaluation converged faster, which translated to fewer iterations.

Table 4.19 Data proportion of the stable cutting and unstable cutting samples

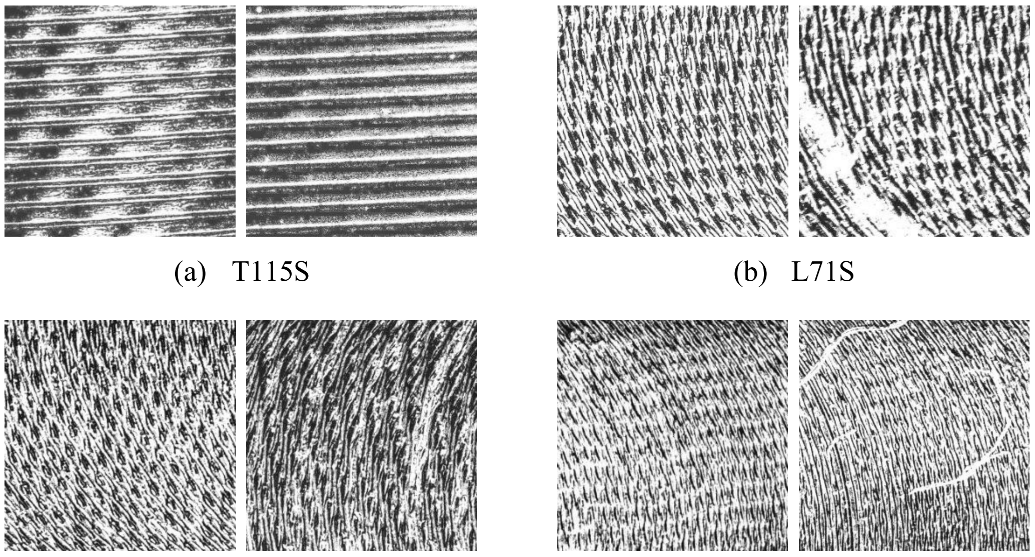
Datasets	**Label**	**Training set**			**Test set**		
		Case 1	**Case 2**	**Case 3**	**Case 1**	**Case 2**	**Case 3**
Turning	unstable cutting	12.38%	12.38%	12.38%	30%	30%	30%
	stable cutting	87.62%	87.62%	87.62%	70%	70%	70%
Slot milling	unstable cutting	25.71%	23.81%	24.76%	30%	50%	40%
	stable cutting	64.76%	66.67%	65.71%	70%	50%	60%
Side milling	unstable cutting	34.29%	35.24%	36.19%	50%	40%	30%
	stable cutting	63.81%	62.86%	61.90%	50%	60%	70%

Table 4.20 Unstable cutting evaluation error E_{ch} for each category of the test datasets.

Datasets	**Label**	**Separate models**			**Combined models**		
		Case 1	**Case 2**	**Case 3**	**Case 1**	**Case 2**	**Case 3**
Turning	unstable cutting	0%	0%	0%	18.33%	9%	2%
	stable cutting	1.07%	0%	0%	0%	0%	0%
Slot milling	unstable cutting	25.00%	18.50%	12.50%	15.83%	14.50%	15.00%
	stable cutting	2.14%	0%	0.83%	1.43%	0.50%	0%
Side milling	unstable cutting	1.00%	59.38%	3.33%	9.00%	61.25%	7.50%
	stable cutting	2.00%	6.67%	12.50%	0.00%	5.83%	14.29%

Further analysis was conducted to point out the source of prediction error on unstable cutting marks identification. Table 4.19 shows the data proportion of the stable and unstable cutting samples in both training and testing cases. Due to the limited available unstable cutting samples, the training sets have imbalanced data with more stable cutting data than the unstable cutting ones. Subsequently, the unstable cutting evaluation error was broken down to each label. Table 4.20 shows the proportion of the unstable cutting evaluation error E_{ch} of each test datasets label. The analysis was done in prediction models with the highest unstable cutting evaluation error.

The results indicated that the errors were more likely to occur on the unstable cutting samples (i.e., mistaking the unstable cutting sample as a stable cutting sample). This is because fewer unstable cutting samples were used for the training, causing the models to lean towards detecting stable cutting samples. In addition, the results indicated that there were a few samples that had a high error rate, termed as 'major error'. Figure 4.21 shows some samples with a major error. All of these samples had unstable cutting label.


(a) T115S (b) L71S

(c) L70S (d) L104A

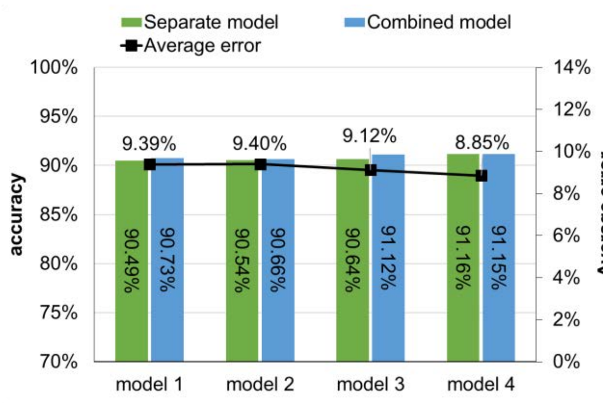
Figure 4.21 Samples with high error rate of unstable cutting evaluation

The detailed error analysis per sample indicates that the major error mainly stemmed from the input data quality. Figure 4.21 (a) depicts two images of an unstable cutting sample (ID T115S). Due to the lighting, the camera could not capture the unstable cutting marks in the right side image; thus the two images appeared to be different. Because of this, the prediction model classified the second image as a stable cutting sample. Meanwhile, Fig. 4.21 (b–d) depicts unstable cutting samples with defects (scratch and dirt) in the second set of images. These defects prompted the prediction models classify them incorrectly. To solve this problem, the quality and quantity of the data should be improved, so that the model could learn how to handle a sample with defects.

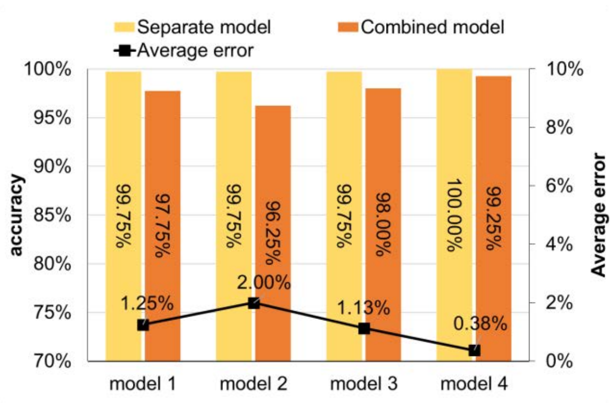
4.2.5 Combined Model Performance

4.2.5.1 Separate vs. Combined Model Comparison

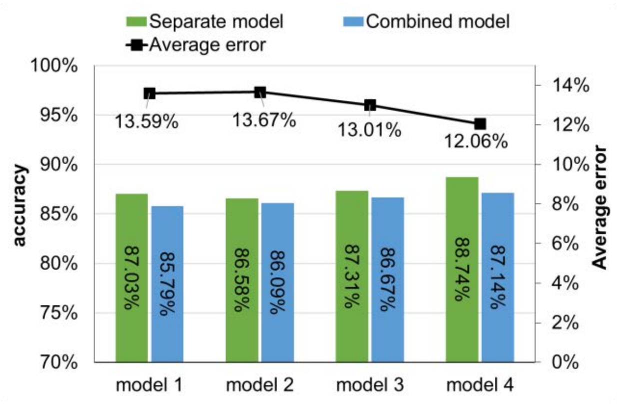
The combined models were trained with the same architecture and parameter settings as the separate models. The main objective of the combined model development was to reduce the training time while maintaining accuracy comparable to that of the separate models. Figure 4.22 depicts the accuracy comparison between the prediction results obtained using the separate and combined models. The bars show the accuracy of separate and combined models, while the marked line shows the average error of both the models. The result confirms the previous hypothesis that the deeper network with a larger input size will have a smaller error. The detailed results of the combined *Ra*- unstable cutting evaluation models are presented in Appendix C.



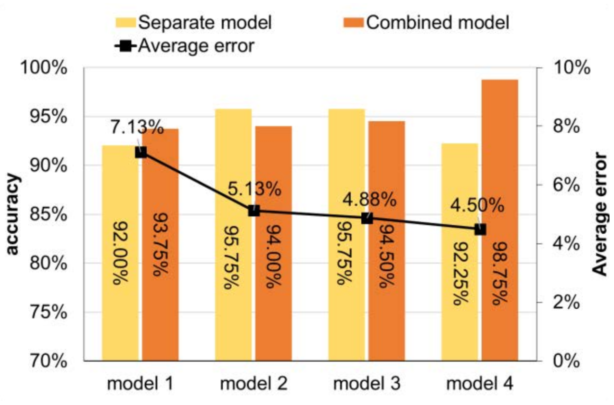
(a) *Ra* prediction accuracy (turned surfaces)



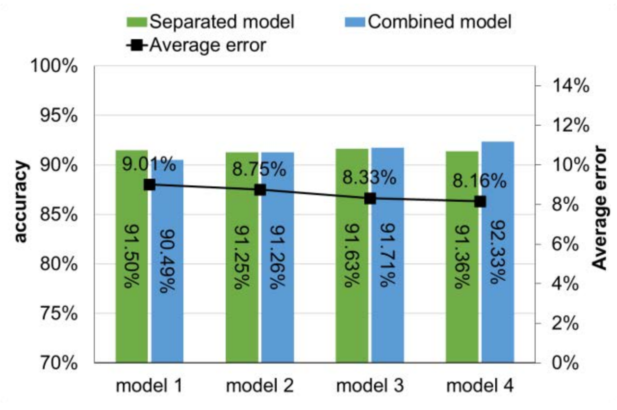
(b) Unstable cutting evaluation accuracy (turned surfaces)



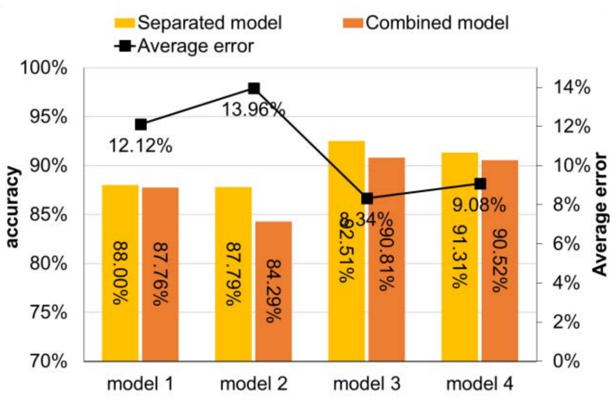
(c) *Ra* prediction accuracy (slot milled surfaces)



(d) Unstable cutting evaluation accuracy (slot milled surfaces)



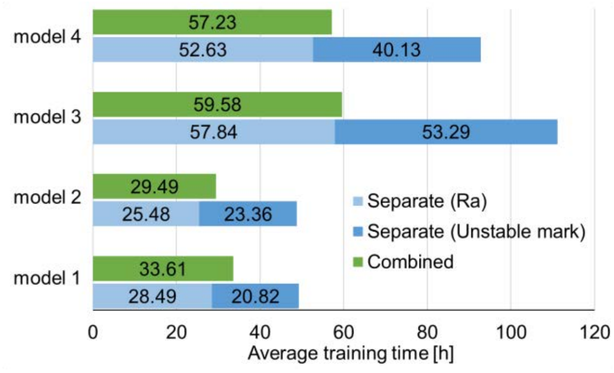
(e) *Ra* prediction accuracy (side milled surfaces)



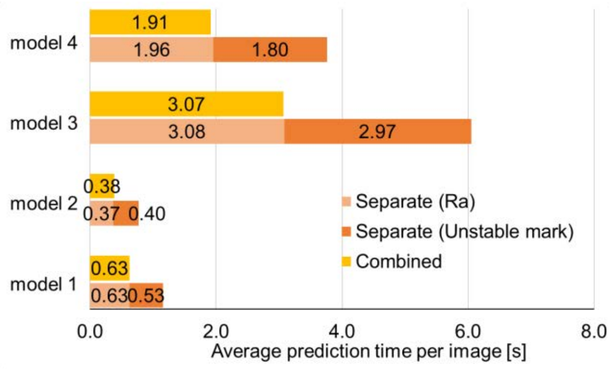
(f) Unstable cutting evaluation accuracy (side milled surfaces)

Figure 4.22 Accuracy comparison between the separate and combined models

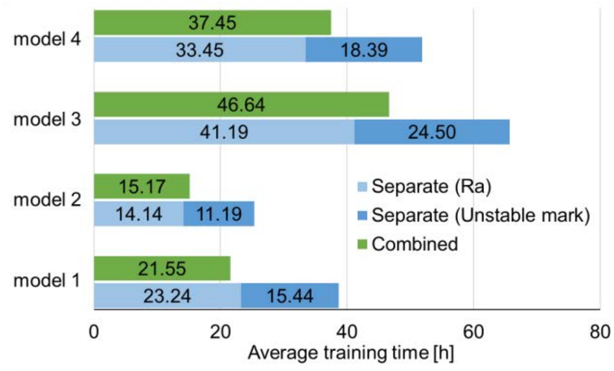
In general, the combined models showed slightly lower accuracy, especially for the unstable cutting prediction problem, than the separate models for the turned surfaces. This is because the target label of the unstable cutting evaluation problem was treated as a continuous numerical value in the combined models. In this case, the classification problem for unstable cutting evaluation was converted to a regression problem. Although it showed lower accuracy, the advantage of the combined model was that it saved almost half of the training and prediction times, as depicted in Fig. 4.23. The processing time reduction was achieved by the combined models because only one model was developed instead of two models with separate systems being used for surface roughness and unstable cutting evaluation.



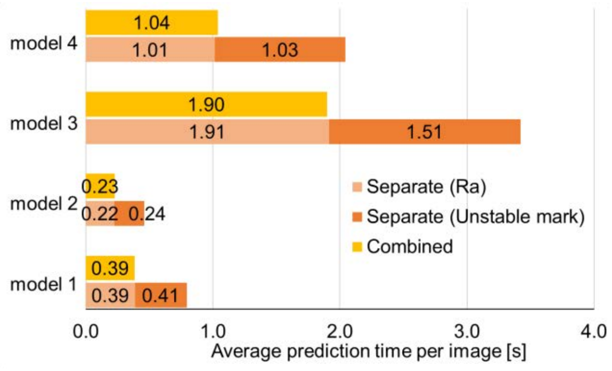
(a) Training time (turned surfaces)



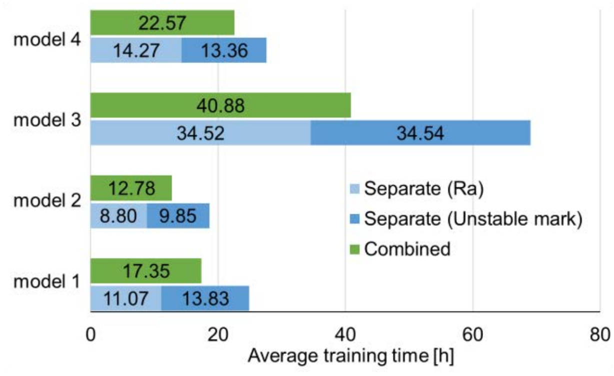
(b) Prediction time (turned surfaces)



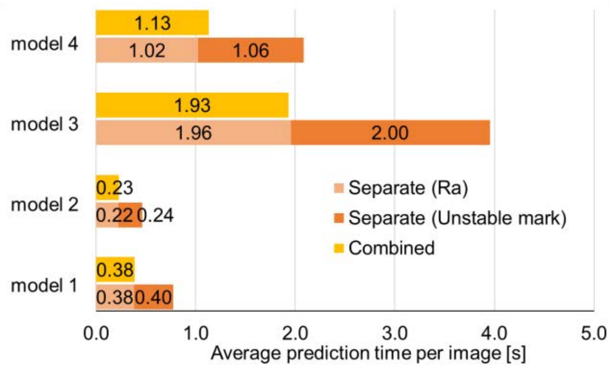
(c) Training time (slot milled surfaces)



(d) Prediction time (slot milled surfaces)

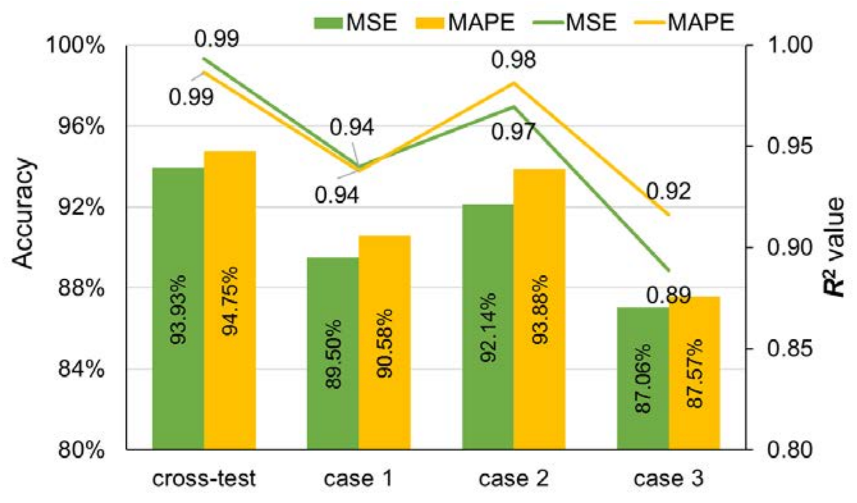


(e) Training time (side milled surfaces)

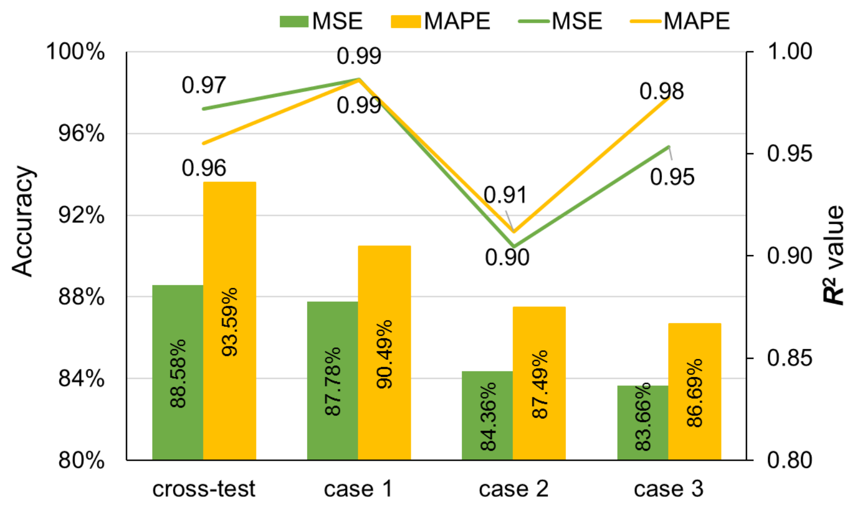


(f) Prediction time (side milled surfaces)

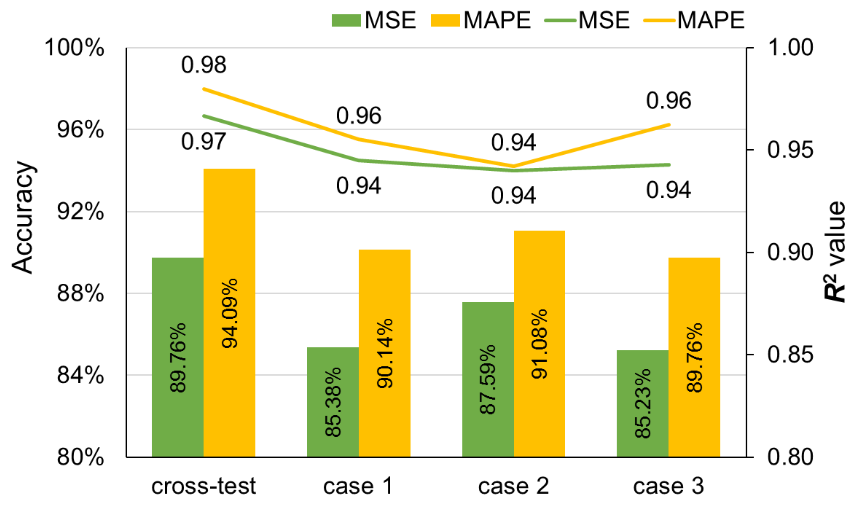
Figure 4.23 Training and prediction time



(a) Turned surface



(b) Slot milled surface



(c) Side milled surface

Figure 4.24 Performance of MSE vs MAPE combined models in turning, slot milling, and side milling data

4.2.5.2 MSE vs. MAPE Models

As indicated in the experimental results of the separate models, MAPE models have the highest accuracy, overall. Therefore, in the combined model experiment, only MAPE was used as the alternative loss function. Subsequently, MAPE model performance was compared against the standard MSE models. In addition, similar to section 4.2.3.2, the experiments of this section were conducted using architecture model 2. Figure 4.24 presents the comparison of MSE and MAPE models in turning, slot milling, and side milling data. Furthermore, a more detailed comparison is provided in Appendix D.

In line with the results of the separate models, the combined models using MAPE scored better in roughness prediction accuracy than MSE models. The results show that MAPE models outperformed MSE models in all the experimental cases, with an average improvement of 1.04 %, 3.47 %, and 4.27 % in turned, slot milled, and side milled surface data, respectively.

4.2.5.3 Results of the Combined *Ra*-*Rz* and *Ra*-*RzJ* Prediction Models

Combined *Ra*-*Rz* models were developed using network architecture 2 and MAPE loss function which displayed better performance than the other functions. The hyperparameters setting was set constant as in the previous experiments. Since both targets were numerical values, R^2, adjusted-R^2, accuracy A, and MAPE loss function were used as measures of performance. These measures were calculated individually for each target, while the training T and prediction time per sample τ were measured together for both the targets. Table 4.21 and 4.22 details the experimental results for the combined *Ra*-*Rz* and *Ra*-*RzJ* models, respectively. Figure 4.25 shows the average *Ra* accuracy of the three proposed prediction schemes: separate, combined *Ra*- unstable cutting, *Ra*-*Rz*, and *Ra*-*RzJ* evaluation models. These values for comparison were the only results of the prediction models using network architecture number 2.

Table 4.21 Experiment results for the combined *Ra*-*Rz* models

Data	**Case**	***Ra***			***Rz***			***T* [h]**	***τ* [s]**
		R^2	**adj-R^2**	A_{Ra}	**R^2**	**adj-R^2**	A_{Rz}		
Turned surface	cross	0.99	0.99	94.57%	0.97	0.97	94.43%	33.31	0.40
	1	0.94	0.94	91.17%	0.94	0.94	90.79%	34.33	0.41
	2	0.98	0.98	93.65%	0.98	0.98	94.13%	32.82	0.37
	3	0.92	0.92	87.20%	0.91	0.91	88.50%	32.63	0.38
Slot milled surface	cross	0.95	0.95	92.56%	0.95	0.95	93.67%	13.06	0.23
	1	0.98	0.98	89.74%	0.98	0.98	88.13%	16.74	0.23
	2	0.94	0.94	85.79%	0.96	0.96	89.12%	11.05	0.23
	3	0.96	0.96	85.40%	0.97	0.97	88.12%	15.87	0.24
Side milled surface	cross	0.98	0.98	93.90%	0.98	0.98	94.37%	8.28	0.24
	1	0.97	0.97	91.98%	0.96	0.96	91.55%	9.83	0.26
	2	0.96	0.96	90.00%	0.96	0.96	90.33%	11.15	0.24
	3	0.94	0.94	88.64%	0.93	0.93	88.49%	7.68	0.27

Table 4.22 Experiment results for the combined *Ra-RzJ* models

Data	**Case**	***Ra***			***RzJ***			***T* [h]**	***τ* [s]**
		R^2	**adj-*R***^{2}	A_{Ra}	***R***^{2}	**adj-*R***^{2}	A_{RzJ}		
Turned surface	cross	0.99	0.99	94.71%	0.97	0.97	93.72%	31.30	0.37
	1	0.95	0.95	92.17%	0.98	0.98	89.37%	33.07	0.37
	2	0.98	0.98	93.58%	0.97	0.97	88.10%	31.28	0.37
	3	0.91	0.90	85.52%	0.92	0.92	83.80%	32.88	0.37
Slot milled surface	cross	0.94	0.94	93.14%	0.93	0.92	92.12%	15.91	0.23
	1	0.99	0.99	89.79%	0.97	0.97	85.38%	16.05	0.24
	2	0.92	0.92	86.02%	0.95	0.95	83.85%	14.77	0.24
	3	0.98	0.98	86.11%	0.91	0.91	80.20%	13.89	0.23
Side milled surface	cross	0.98	0.98	94.06%	0.94	0.94	92.43%	18.26	0.24
	1	0.97	0.97	92.93%	0.88	0.88	87.30%	13.01	0.24
	2	0.94	0.94	89.86%	0.91	0.91	90.32%	19.40	0.23
	3	0.94	0.94	88.27%	0.82	0.82	80.66%	18.54	0.23

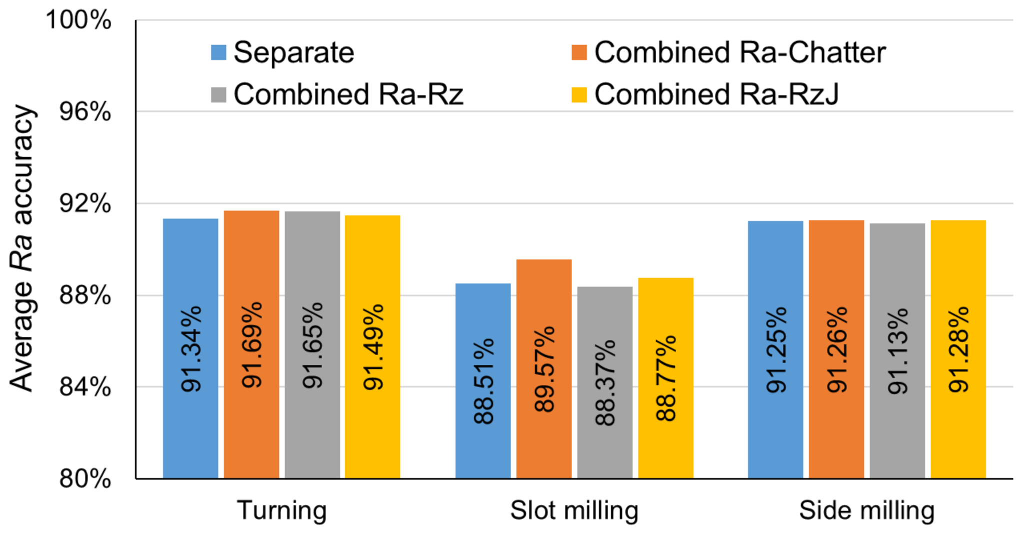


Figure 4.25 Average *Ra* accuracy of the proposed prediction models

As depicted in Fig. 4.25, the use of the combined models did not significantly affect the roughness prediction accuracy, thus confirming the results of section 4.2.5.1. Furthermore, the results indicated that the *Rz* prediction accuracy was in line with and comparable to *Ra* accuracy, as the average accuracies for *Ra* and *Rz* were 90.38 % and 90.97 %, respectively in all the datasets. However, there was a disparity between the *Ra* and *RzJ* accuracies; *RzJ* accuracy was significantly lower than both the *Ra* and *Rz* accuracies. The average *RzJ* accuracy was 87.27 %, while the *Ra* accuracy obtained by the same combined model was 90.51 %. This discrepancy was probably caused by the difficulties in quantifying the effect of the defects. In the machined surface, scratches often left a thin line of deep valley. *RzJ* was much more significantly influenced by those defects (scratch) than either *Ra* or *Rz*. As is true for the case of unstable cutting, *Ra*, and *Rz* predictions, the prediction models were designed to identify the ridge-valley patterns by analyzing the texture feature. Although the scratch line significantly altered the *RzJ* values, the prediction model was less cognizant of its effect. In such a condition, the development of a prediction model to exclusively detect the defects might result in a better evaluation of the machined surface. Furthermore, the individual accuracy for *Ra*, *Rz*, and *RzJ* predictions in all the datasets were investigated. The combined *Ra*-*Rz* model accuracy in each case is shown in Fig. 4.26.

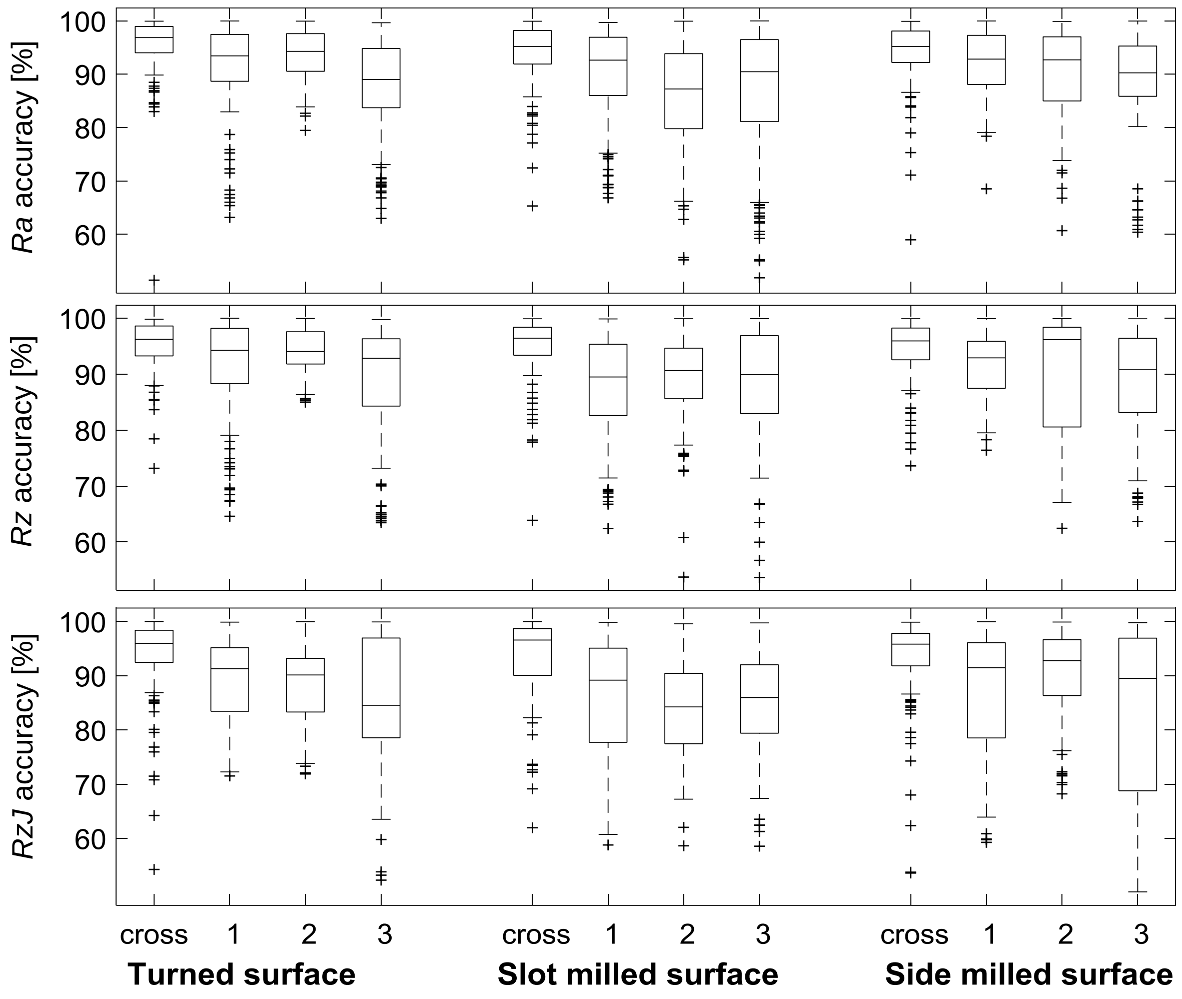


Figure 4.26 Boxplot of individual accuracies of *Ra*, *Rz*, and *RzJ* in the combined *Ra*-*Rz* prediction model

The box represents the values between 1st and 3rd quartiles with the maximum whisker length specified as 1.0 times the interquartile range, while the middle line denotes the median. The boxplots indicate that there are outliers with low prediction accuracy. Those outliers are more prominent in the slot and side milled data since the ridge-valley pattern in both surfaces was less clear than in the turned surface data. Moreover, the outliers maybe caused by defects, such as cracks, breaks, scratches, and dirt on the surface. To minimize such effects, taking and evaluating multiple images of a surface, and then aggregating the predicted values is advocated.

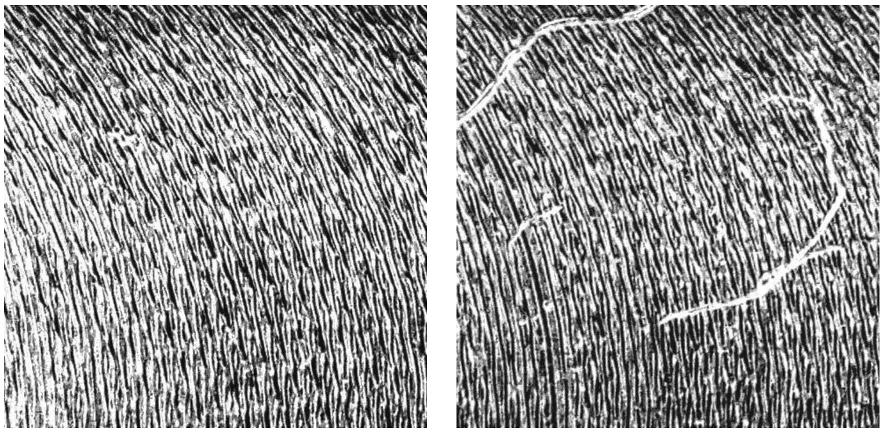
4.2.6 Analysis and Discussion

Experimental studies were carried out for turning and milling data with a relatively wide roughness range. Therefore, the developed models also had a wide application range, which depended on the range in the training data. The developed models were built based on two materials, S45C and A5052. Further analysis showed that there was no discrepancy between the two materials, and that the models worked well for different types of materials. Based on these results, roughness and unstable cutting evaluations for different materials may be possible, given that the new materials have properties similar to either S45C or A5052, and have no extra patterns that would distort the prediction process.

This study attempts to develop general prediction models that cover different types of material. Hence, S45C carbon steel and A5052 aluminum alloy specimens were used for the experiment. In the literature, most studies only developed prediction models specific to certain material. Developing a general prediction model is comparatively more difficult than a model for a single type of material because each material has a different light reflection coefficient. This difference affects the vision-based system because the system depends on the light intensity in the image. This problem may trouble models that are based on the gray-level average, owing to their dependence on the intensities in the image. On the other hand, the use of deep learning and selected image processing in this study provides an advantage, because it relies more on the pattern in the image than the intensity levels. Therefore, there is no significant difference between the performances of the single-material models and our multi-material model.

The roughness estimation accuracy ranges were 86 % – 96 % for turned surfaces, 81 % – 93 % for slot milled surfaces, and 88 % – 94 % for side milled surfaces. Furthermore, the results depicted in Fig. 4.18 and 4.19 confirm the presumption that the use of deeper networks improved the prediction accuracy albeit at the expense of increased processing time. Overall, architectures 3 and 4 appeared to give better results due to their larger input sizes and deeper networks. However, considering both accuracy and processing time, architecture 2 provided more optimum effectiveness-efficiency ratio. In the unstable cutting evaluation, the proposed method was able to obtain high identification accuracy in turning data. On the other hand, less visible unstable cutting marks in the milling surface images compounded the prediction process which resulted in a lower accuracy. On the whole, the average accuracies of prediction models for the turning datasets were higher than those for the milled surfaces, because the turned surfaces had simple and clear patterns of ridge-valley, and less surface defects. Therefore, it was easier to identify their features. In addition, the sample size of training data for turning dataset was considerably larger than the milling datasets so that it helped the networks to learn correct features and to avoid over-fitting.

In surface roughness estimation problem, the errors might be caused by image quality and dirt, as shown in Fig. 4.27. The figure shows two images of a sample (L104A) with a measured *Ra* = 0.71 μm. Image (a) which shows a clean surface returned a predicted *Ra* of 0.61 μm — a prediction accuracy of 86.29 %. Meanwhile, Image (b) which contained dirt returned a predicted *Ra* of 0.54 μm — a prediction accuracy of only 76.96 %. The noises caused by image quality and surface defects created prediction outliers with high errors. Similar to the optical/laser measurements, the proposed vision method is also affected by cutting fluid and dirt. Therefore, the vision method will work best for a clean surface.



(a) clean surface
predicted *Ra* = 0.61 μm

(b) surface with dirt
predicted *Ra* = 0.54 μm

Figure 4.27 Effect of dirt on the machined surfaces toward *Ra* prediction

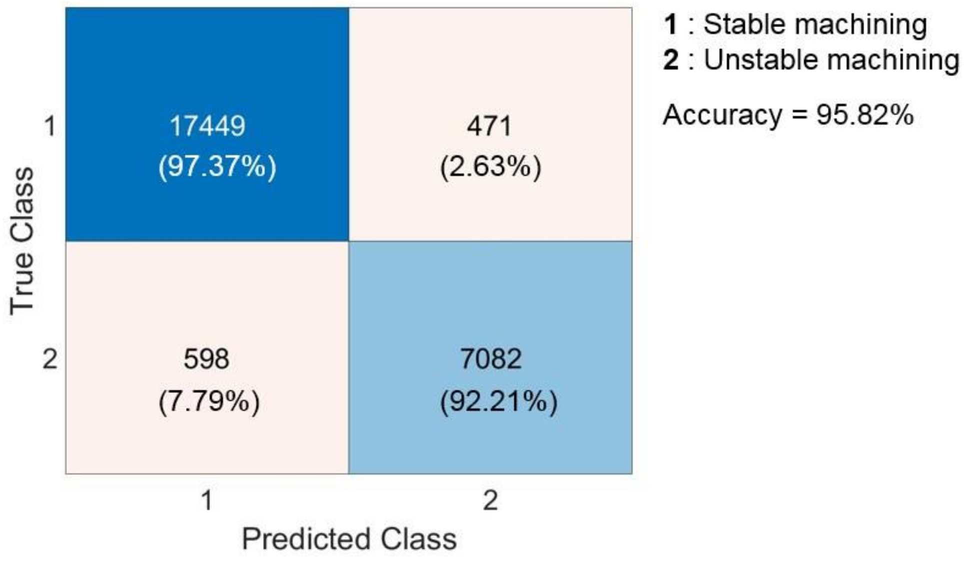


Figure 4.28 Confusion matrix of unstable cutting mark identification

In the vibration identification problem, the error rate of unstable cutting sample was slightly higher than that of the stable cutting sample, as shown in the confusion matrix of Fig. 4.28. This was probably caused by a disparity in the number of samples between the two classes. The stable cutting class had significantly more images than the unstable cutting class. Therefore, the prediction models might lean more towards predicting a stable cutting condition.

The proposed combined model that simultaneously predicted both surface roughness and unstable cutting marks gave results comparable to those of the individual prediction models. Therefore, it was an effective method for reducing the computation time. In both the surface roughness and unstable cutting evaluation problems, texture features were important for generating discriminated representations. Because texture features were prominent, several adaptations have been applied when building deep learning models, such as the use of average pooling, and minimizing the pooling process, to retain the texture.

The advantages of the proposed method lie in its ease and capability for fast prediction with acceptable accuracy. Therefore, this method is desirable for quick inspection in-between the machining processes. Another important benefit of the proposed system is that it only uses standard room lighting and does not require a special illumination setting. However, owing to the lack of special lighting, non-uniform intensity may be obtained on the surface of a machined specimen. This problem is compensated by dividing the image into small regions and then performing histogram equalization with various intensity levels.

For the unstable cutting marks identification, although the vision method cannot be applied for real-time monitoring unlike the cutting forces or sound emission analysis, it has the advantage of easier development and application. Chatter frequencies are affected by the dynamic characteristics of the machine tool. Hence, the analysis based on the cutting forces and sound emission might differ among the machines, and a specific identification model should be developed for each machine. The vision-based method directly analyzes the unstable cutting marks on the surface. It is more general, practical, and easier to be developed and applied to various machines. Another advantage of the proposed method is that it is less reliant on the complex illumination requirement. It is particularly useful in practical implementations since installing excessive lighting often is hindered due to the confined space inside the machine. Furthermore, it can conserve the energy required for the illumination setup.

Owing to its convolutional process and deep architecture characteristics, the proposed prediction models using the CNN took relatively longer training time than the other vision-based prediction methods. However, this longer training time can be compensated by omitting the feature extraction which is a mandatory step in the other methods. Thus, the prediction and the testing time can be reduced. In view of its performance, the proposed system using the CNN offers a competitive alternative for surface quality inspection method.

In general, the proposed system only considers the surface roughness and unstable cutting marks as the prediction target. Therefore, the current method omits other components of surface texture, such as waviness. Although waviness is less commonly used than roughness as an indicator of the surface texture, it still holds an important role in checking the instabilities in machining. The waviness can also be used to indicate vibration in the machined surface. However, since the wavelength of waviness is higher than that of roughness, a larger area of the machined surface must be analyzed. In the vision method, this requirement could be an issue since capturing a large area may result in a less detailed ridge-valley pattern. On the other hand, if the level of detail of the ridge-valley pattern is maintained, then a larger image resolution is required which results in heavy data and long prediction process. Hence, the vision-based prediction method for roughness and waviness might be more economical to be separately developed.

The absolute values of training T and prediction time τ depend on the specification of the computer. Therefore, their values might change if the proposed models are applied in a different hardware. Nevertheless, the results of the training and the prediction times provided a perspective of the network model performance comparison. Furthermore, it outlined the effect of the architecture, combined as well as separate systems, loss function, and hyperparameters setting on training and prediction times.

CHAPTER 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this study, the possibility of applying computer vision for addressing two issues related to machining processes in the manufacturing field is evaluated. In the first problem, machine vision is applied for tool-type recognition, as well as estimation of tool-dimensions and overhang. The aim of the system is to assist a machine operator during the tool setting, and thus automate the process. The proposed CNN models, based on deep learning have been proved to achieve a high accuracy. The models are extended for developing R-CNNs to localize the exact position of the tool in the image. Edge detection is then applied to the detected ROI to identify the edges of the tool and the holder. Subsequently, a set of algorithms is used to estimate the tool dimension and overhang. The information related to both the tool recognition and the dimension estimation is used to automatically generate an NC program for the dimension measurement process using a contact-based displacement sensor. The results obtained indicate that the proposed system offers a competitive method for automatic tool recognition with considerably low processing time.

In the second problem, the CNN prediction models have been developed for vision-based surface roughness estimation and identification of unstable cutting marks. The use of vision-based prediction with deep learning has several advantages. It provides quick and easy setup, and non-invasive measurement compared with the time-consuming profilometer. Unlike the other vision-based methods, the proposed system obviates the need for manual feature extraction process, special illumination, and excessive image processing. Consequently, the prediction time is further reduced. With an average prediction time of less than four seconds per sample, the proposed method is suitable for application as an on-machine, quick measurement alternative.

5.2 Future Research

5.2.1 Limitations of the Current Models

As is true of the other supervised prediction methods, the usage of the deep learning models is also limited to predictions for samples with characteristics that have been learned. In the tool type identification, the prediction model can only predict the tool type that has been learned during the training process. If a completely new tool type is introduced, then the prediction model may recognize it as one of the learned types that is closest in terms of the shape.

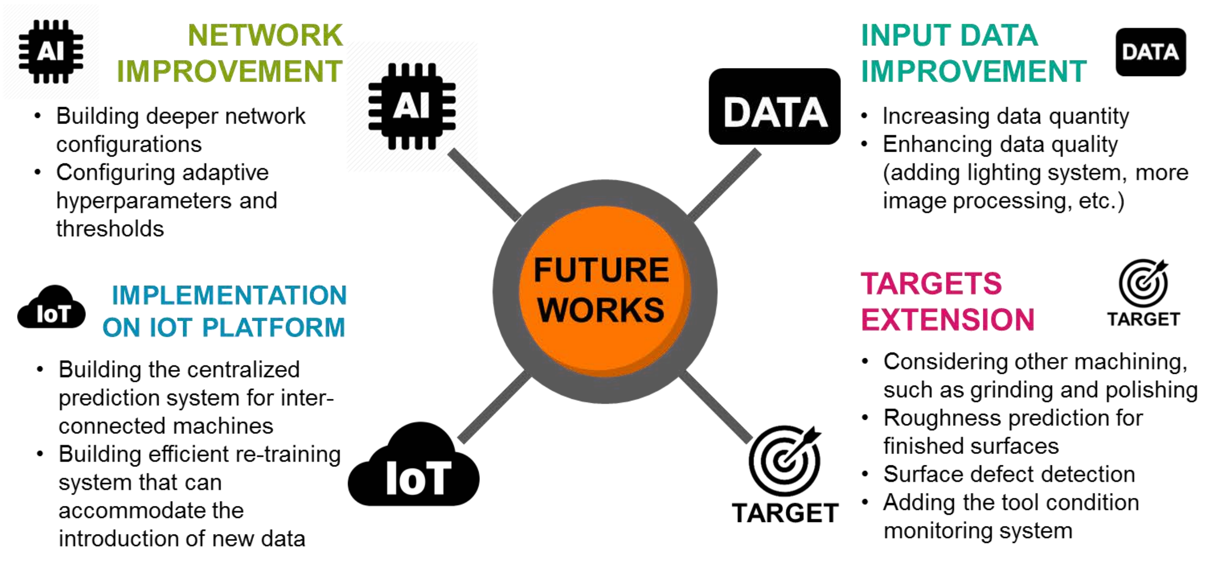
In surface roughness estimation, the prediction models are limited to the roughness range of the training data, i.e., *Ra* in the range of 0.26 – 8.05 μm for slot milling data, or for the corresponding rough cut surface. The prediction for a sample with *Ra* outside this range may be inaccurate. It is possible to develop the vision-based prediction model for finished cut specimens as long as the pattern of 'ridge-valley' is visible. However, to create prediction models for finished cut surfaces, higher magnification is required to capture the finer patterns of 'ridge-valley'. To capture such images of the finished surfaces with lower *Ra*, more sophisticated devices, such as high magnification macro lens or scanning electron microscope (SEM), are required. Extra illumination might be necessary to enhance the difference between ridge and valley of the finished surfaces.

5.2.2 Future Improvements of the System

Further improvements for both problems can span several directions, as illustrated in Fig. 5.1. The improvement on the prediction model can be achieved by modifying the network configuration, for example, by adding more weighted layers and using adaptive hyperparameter setting. The issue of setting hyperparameter values is an interesting topic to explore as some values are pre-determined by the developer. Setting the deep learning hyperparameter values requires the expertise of the developer, and often requires 'trial and error' to find the optimum setting. By developing an adaptive system, it is expected that the deep learning system can adjust those values automatically during the training process based on the current cost function. With an adaptive system, the application and implementation of AI in the manufacturing field would be more realistic. An adaptive system can also be developed to adjust the confidence threshold of the R-CNN to minimize the number of predictions of false positives and false negatives. In addition, other algorithms such as a single shot multibox detector (SSD) can be considered for future studies, to obtain a more accurate ROI.

On the input data aspect, improvements can be achieved by enhancing both the quality and the quantity of the training data. The quantity and quality enhancement can be achieved by increasing the sample size and/or the image size, performing more image processing, using more accurate profilometer data as the input, using more precise image acquisition device, and adding special illumination. However, it should be noted that those improvements can be achieved only at the expense of longer processing time and a more complex setup.

With regard to industry 4.0, the integration of internet of things (IoT) and AI allows the building of intelligent systems and smart manufacturing. In our problem case studies, the inter-connection of the machines and the devices helps in acquiring 'big data' that is essential for deep learning. Moreover, it provides an infrastructure for centralized control, so that the AI models for prediction are not individually stored in each machine but in a central computer that controls the machines. With more information acquired from the connected machines, it is possible to extend the prediction models and overcome their limitations. In the cutting tool identification, more tool types can be accommodated, and the prediction and measurement processes can be monitored remotely. In the surface quality inspection, the prediction models may cover other materials and expand the surface roughness range that can be predicted.



NETWORK IMPROVEMENT
• Building deeper network configurations
• Configuring adaptive hyperparameters and thresholds
AI
DATA
INPUT DATA IMPROVEMENT
• Increasing data quantity
• Enhancing data quality (adding lighting system, more image processing, etc.)
FUTURE WORKS
IMPLEMENTATION ON IOT PLATFORM
IoT
• Building the centralized prediction system for inter-connected machines
• Building efficient re-training system that can accommodate the introduction of new data
TARGETS EXTENSION
TARGET
• Considering other machining, such as grinding and polishing
• Roughness prediction for finished surfaces
• Surface defect detection
• Adding the tool condition monitoring system

Figure 5.1 Directions for future work

Finally, another potential improvement is the target extension, for example, for surface defect detection and considering other machined surfaces. Other measures of roughness such as Rp and Rq, or waviness parameters, such as Wa and Wt, can be added as indicators of surface quality. Future studies may also cover other types of materials and machining processes, such as ball-end milling, polishing and grinding. The ground and polished surfaces may have random pattern of ridge-valley which can be challenging for the network to identify the feature. Furthermore, the cutting conditions available, namely cutting speed C, spindle speed S, feed rate F, and depth of cut D can be considered as secondary inputs for improving the prediction accuracy. For the tool identification problem, the system can be extended for evaluation of tool geometries, for example, identifying the rake angle, flank angle, clearance angle, helix angle, and number of flutes. The evaluation method for this problem can be developed by using edge detection and estimation algorithms, similar to the estimation of the tool dimensions. Finally, the vision methods based on deep learning can also be developed for tool condition monitoring to detect tool wear and broken tools. In addition, the tool wear area can be estimated by using edge detection and estimation algorithm, similar to the proposed method for tool dimension estimation.

BIBLIOGRAPHY

[1] Lucke, D., Constantinescu, C., and Westkämper, E., Smart factory – a step towards the next generation of manufacturing, Manufacturing Systems and Technologies for the New Frontier, (2008), pp. 115–118.

[2] Thames, L. and Schaefer, D., Software-defined cloud manufacturing for Industry 4.0, Procedia CIRP, Vol. 52, (2016), pp. 12–17.

[3] Lasi, H., Fettke, P., Kemper, H.G., Feld, T., and Hoffmann, M., Industry 4.0, Business & Information Systems Engineering, Vol. 6, No. 4, (2014), pp. 239-242.

[4] Lee, J., Davari, H., Singh, J., and Pandhare, V., Industrial Artificial Intelligence for industry 4.0-based manufacturing systems, Manufacturing Letters, Vol. 18, (2018), pp. 20-23.

[5] Oztemel, E., Intelligent manufacturing systems, Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management, (2010), pp. 1-41.

[6] Kaplan, A. and Haenlein, M., Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence, Business Horizons, Vol. 62, No. 1, (2019), pp. 15-25.

[7] Sadeghian, R. and Sadeghian, M.R., A decision support system based on artificial neural network and fuzzy analytic network process for selection of machine tools in a flexible manufacturing system, The International Journal of Advanced Manufacturing Technology, Vol. 82, No. 9-12, (2016), pp. 1795-1803.

[8] Nguyen, H.T., Dawal, S.Z.M., Nukman, Y., Rifai, A.P., and Aoyama, H., An integrated MCDM model for conveyor equipment evaluation and selection in an FMC based on a fuzzy AHP and fuzzy ARAS in the presence of vagueness, PloS One, Vol. 11, No. 4, (2016), e0153222.

[9] Zhang, X., Liang, Y., and Zhou, J., A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM, Measurement, Vol. 69, (2015), pp. 164-179.

[10] Yin, Z. and Hou, J., Recent advances on SVM based fault diagnosis and process monitoring in complicated industrial processes, Neurocomputing, Vol. 174, (2016), pp. 643-650.

[11] Costamagna, P., De Giorgi, A., Magistri, L., Moser, G., Pellaco, L., and Trucco, A., A classification approach for model-based fault diagnosis in power generation systems based on solid oxide fuel cells, IEEE Transactions on Energy Conversion, Vol. 31, No. 2, (2016), pp. 676-687.

[12] Zhang, C. and Zhang, H., Modelling and prediction of tool wear using LS-SVM in milling operation, International Journal of Computer Integrated Manufacturing, Vol. 29, No. 1, (2016), pp.76-91.

[13] Pandiyan, V., Caesarendra, W., Tjahjowidodo, T., and Tan, H.H., In-process tool condition monitoring in compliant abrasive belt grinding process using support vector machine and genetic algorithm, Journal of Manufacturing Processes, Vol. 31, (2018), pp. 199-213.

[14] Rodan, A., Sheta, A.F., and Faris, H., Bidirectional reservoir networks trained using SVM + privileged information for manufacturing process modeling, Soft Computing, Vol. 21, No. 22, (2017), pp. 6811-6824.

[15] Khorasani, A. and Yazdi, M.R.S., Development of a dynamic surface roughness monitoring system based on artificial neural networks (ANN) in milling operation, The International Journal of Advanced Manufacturing Technology, Vol. 93, No. 1-4, (2017), pp. 141-151.

[16] Palani, S. and Natarajan, U., Prediction of surface roughness in CNC end milling by machine vision system using artificial neural network based on 2D Fourier transform, The International Journal of Advanced Manufacturing Technology, Vol. 54, No. 9-12, (2011), pp. 1033-1042.

[17] Samtaş, G., Measurement and evaluation of surface roughness based on optic system using image processing and artificial neural network, The International Journal of Advanced Manufacturing Technology, Vol. 73, No. 1-4, (2014), pp. 353-364.

[18] Li, L., Liu, F., Chen, B., and Li, C.B., Multi-objective optimization of cutting parameters in sculptured parts machining based on neural network, Journal of Intelligent Manufacturing, Vol. 26, No. 5, (2015), pp. 891-898.

[19] D'Addona, D.M., Ullah, A.S., and Matarazzo, D., Tool-wear prediction and pattern-recognition using artificial neural network and DNA-based computing. Journal of Intelligent Manufacturing, Vol. 28, No. 6, (2017), pp. 1285-1301.

[20] Hesser, D.F. and Markert, B., Tool wear monitoring of a retrofitted CNC milling machine using artificial neural networks, Manufacturing Letters, Vol. 19, (2019), pp. 1-4.

[21] Nguyen, H.T., Dawal, S.Z.M., Nukman, Y., and Aoyama, H., A hybrid approach for fuzzy multi-attribute decision making in machine tool selection with consideration of the interactions of attributes, Expert Systems with Applications, Vol. 41, No. 6, (2014), pp. 3078-3090.

[22] Su, T.S. and Lin, Y.F., Fuzzy multi-objective procurement/production planning decision problems for recoverable manufacturing systems, Journal of Manufacturing Systems, Vol. 37, (2015), pp. 396-408.

[23] Gupta, M.K., Sood, P.K., and Sharma, V.S., Machining parameters optimization of titanium alloy using response surface methodology and particle swarm optimization

under minimum-quantity lubrication environment, Materials and Manufacturing Processes, Vol. 31, No. 13, (2016), pp. 1671-1682.

[24] Loganathan, M.K. and Gandhi, O.P., Maintenance cost minimization of manufacturing systems using PSO under reliability constraint, International Journal of System Assurance Engineering and Management, Vol. 7, No. 1, (2016), pp. 47-61.

[25] Rifai, A.P., Dawal, S.Z.M., Zuhdi, A., Aoyama, H., and Case, K., Reentrant FMS scheduling in loop layout with consideration of multi loading-unloading stations and shortcuts, The International Journal of Advanced Manufacturing Technology, Vol. 82, No. 9-12, (2016), pp. 1527-1545.

[26] Rifai, A.P., Nguyen, H.T., and Dawal, S.Z.M., Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling, Applied Soft Computing, Vol. 40, (2016), pp. 42-57.

[27] Rifai, A.P., Nguyen, H.T., Aoyama, H., Dawal, S.Z. M., and Masruroh, N.A., Non-dominated sorting biogeography-based optimization for bi-objective reentrant flexible manufacturing system scheduling, Applied Soft Computing, Vol. 62, (2018), pp. 187-202.

[28] Ma, G. and Zhang, F., Genetic algorithms for manufacturing process planning, Variants of Evolutionary Algorithms for Real-World Applications, (2012), pp. 205-244.

[29] Deng, Z., Zhang, H., Fu, Y., Wan, L., and Lv, L., Research on intelligent expert system of green cutting process and its application, Journal of Cleaner Production, Vol. 185, (2018), pp. 904-911.

[30] Torres-Treviño, L.M., Escamilla-Salazar, I.G., González-Ortíz, B., and Praga-Alejo, R., An expert system for setting parameters in machining processes, Expert Systems with Applications, Vol. 40, No. 17, (2013), pp. 6877-6884.

[31] Syn, C Z., Mokhtar, M., Feng, C.J., and Manurung, Y.H., Approach to prediction of laser cutting quality by employing fuzzy expert system, Expert Systems with Applications, Vol. 38, No. 6, (2011), pp. 7558-7568.

[32] Vundavilli, P.R., Parappagoudar, M.B., Kodali, S.P., and Benguluri, S., Fuzzy logic-based expert system for prediction of depth of cut in abrasive water jet machining process, Knowledge-Based Systems, Vol. 27, (2012), pp. 456-464.

[33] Jones, B., Jenkinson, I., Yang, Z., and Wang, J., The use of Bayesian network modelling for maintenance planning in a manufacturing industry, Reliability Engineering & System Safety, Vol. 95, No. 3, (2010), pp. 267-277.

[34] Yang, L. and Lee, J., Bayesian belief network-based approach for diagnostics and prognostics of semiconductor manufacturing systems, Robotics and Computer-Integrated Manufacturing, Vol. 28, No. 1, (2012), pp. 66-74.

[35] Boutros, T. and Liang, M., Detection and diagnosis of bearing and cutting tool faults using hidden Markov models, Mechanical Systems and Signal Processing, Vol. 25, No. 6, (2011), pp. 2102-2124.

[36] Geramifard, O., Xu, J.X., Zhou, J.H., and Li, X., A physically segmented hidden Markov model approach for continuous tool condition monitoring: Diagnostics and prognostics, IEEE Transactions on Industrial Informatics, Vol. 8, No. 4, (2012), pp. 964-973.

[37] Abellan-Nebot, J.V., and Subirón, F.R., A review of machining monitoring systems based on artificial intelligence process models, The International Journal of Advanced Manufacturing Technology, Vol. 47, No. 1-4, (2010), pp. 237-257.

[38] Bai, Y., Li, C., Sun, Z., and Chen, H., Deep neural network for manufacturing quality prediction, IEEE Prognostics and System Health Management Conference, (2017), pp. 1-5.

[39] Lee, K.B., Cheon, S., and Kim, C.O., A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes, IEEE Transactions on Semiconductor Manufacturing, Vol. 30, No. 2, (2017), pp. 135-142.

[40] Malamas, E.N., Petrakis, E.G., Zervakis, M., Petit, L., and Legat, J.D., A survey on industrial vision systems, applications and tools, Image and Vision Computing, Vol. 21, No. 2, (2003), pp. 171-188.

[41] Ahmad, R., Tichadou, S., and Hascoet, J.Y., Generation of safe and intelligent tool-paths for multi-axis machine-tools in a dynamic 2D virtual environment, International Journal of Computer Integrated Manufacturing, Vol. 29, No. 9, (2016), pp. 982-995.

[42] Mikołajczyk, T., Kłodowski, A., and Mrozinski, A., Camera-Based Automatic System for Tool Measurements and Recognition, Procedia Technology, Vol. 22, (2016), pp. 1035-1042.

[43] Elgargni, M., Al-Habaibeh, A., and Lotfi, A., Cutting tool tracking and recognition based on infrared and visual imaging systems using principal component analysis (PCA) and discrete wavelet transform (DWT) combined with neural networks, The International Journal of Advanced Manufacturing Technology, Vol. 77, No. 9-12, (2015), pp. 1965-1978.

[44] Nakamoto, K. and Takeuchi, Y., Recent Advances in Multiaxis Control and Multitasking Machining, International Journal of Automation Technology, Vol. 11, No. 2, (2017), pp. 140-154.

[45] Jędrzejewski, J. and Kwaśny, W., Discussion of machine tool intelligence, based on selected concepts and research, Journal of Machine Engineering, Vol. 15, (2015), pp. 5-26.

[46] Moriwaki, T., Multi-functional machine tool, CIRP Annals, Vol. 57, No. 2, (2008), pp. 736-749.

[47] Ahmad, R., Tichadou, S., and Hascoet, J.Y., 3D safe and intelligent trajectory generation for multi-axis machine tools using machine vision, International Journal of Computer Integrated Manufacturing, Vol. 26, No. 4, (2013), pp. 365-385.

[48] Mei, K.J. and Lee, R.S., Collision detection for virtual machine tools and virtual robot arms using the Shared Triangles Extended Octrees method, International Journal of Computer Integrated Manufacturing, Vol. 29, No. 4, (2016), pp. 355-373.

[49] Zhang, X., Tsang, W.M., Yamazaki, K., and Mori, M., A study on automatic on-machine inspection system for 3D modeling and measurement of cutting tools, Journal of Intelligent Manufacturing, Vol. 24, No. 1, (2013), pp. 71-86.

[50] Karabagli, B., Simon, T., and Orteu, J.J., A new chain-processing-based computer vision system for automatic checking of machining set-up application for machine tools safety, The International Journal of Advanced Manufacturing Technology, Vol. 82, No. 9-12, (2016), pp.1547-1568.

[51] Liu, J., Lu, E., Yi, H., Wang, M., and Ao, P., A new surface roughness measurement method based on a color distribution statistical matrix, Measurement, Vol. 103, (2017), pp. 165-178.

[52] Shahabi, H.H. and Ratnam, M.M., Noncontact roughness measurement of turned parts using machine vision, The International Journal of Advanced Manufacturing Technology, Vol. 46, No. 1–4, (2010), pp. 275–284.

[53] Shahabi, H.H. and Ratnam, M.M., Prediction of surface roughness and dimensional deviation of workpiece in turning: a machine vision approach, The International Journal of Advanced Manufacturing Technology, Vol. 48, No. 1–4, (2010), pp. 213–226.

[54] Sarma, P.M.M.S., Karunamoorthy, L., and Palanikumar, K., Surface roughness parameters evaluation in machining GFRP composites by PCD tool using digital image processing, Journal of Reinforced Plastics and Composites, Vol. 28, No. 13, (2009), pp. 1567-1585.

[55] Dhanasekar, B. and Ramamoorthy, B., Restoration of blurred images for surface roughness evaluation using machine vision, Tribology International, Vol. 43, No. 1-2, (2010), pp. 268-276.

[56] Kamguem, R., Tahan, S.A., and Songmene, V., Evaluation of machined part surface roughness using image texture gradient factor, International Journal of Precision Engineering and Manufacturing, Vol. 14, No. 2, (2013), pp. 183-190.

[57] Nammi, S. and Ramamoorthy, B., Effect of surface lay in the surface roughness evaluation using machine vision, Optik, Vol. 125, No. 15, (2014), pp. 3954-3960.

[58] Kumar, R., Kulashekar, P., Dhanasekar, B., and Ramamoorthy, B., Application of digital image magnification for surface roughness evaluation using machine vision, International Journal of Machine Tools and Manufacture, Vol. 45, No. 2, (2005), pp. 228-234.

[59] Elango, V. and Karunamoorthy, L., Effect of lighting conditions in the study of surface roughness by machine vision-an experimental design approach, The International Journal of Advanced Manufacturing Technology, Vol. 37, No. 1-2, (2008), pp. 92-103.

[60] Chiou, R.Y., Kwon, Y.J., Tseng, T.L.B., and Mauk, M., Experimental Study of High Speed CNC Machining Quality by Noncontact Surface Roughness Monitoring, International Journal of Mechanical Engineering and Robotics Research, Vol. 4, No. 4, (2015), pp. 282-286.

[61] Bhat, N.N., Dutta, S., Pal, S.K., and Pal, S., Tool condition classification in turning process using hidden Markov model based on texture analysis of machined surface images, Measurement, Vol. 90, (2016), pp. 500-509.

[62] Gadelmawla, E.S., Estimation of surface roughness for turning operations using image texture features, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 225, No. 8, (2011), pp. 1281-1292.

[63] Min, L., Gao, L., Zhang, X., and Wang, Z., Surface roughness measurement based on image texture analysis, IEEE 7th International Congress on Image and Signal Processing (CISP), (2014), pp. 514-519.

[64] Dutta, S., Pal, S.K., and Sen, R., On-machine tool prediction of flank wear from machined surface images using texture analyses and support vector regression, Precision Engineering, Vol. 43, (2016), pp. 34-42.

[65] Bhat, N.N., Dutta, S., Vashisth, T., Pal, S., Pal, S.K., and Sen, R., Tool condition monitoring by SVM classification of machined surface images in turning, The International Journal of Advanced Manufacturing Technology, Vol. 83, No. 9-12, (2016), pp. 1487-1502.

[66] Morala-Argüello, P., Barreiro, J., and Alegre, E., A evaluation of surface roughness classes by computer vision using wavelet transform in the frequency domain, The International Journal of Advanced Manufacturing Technology, Vol. 59, No. 1-4, (2012), pp. 213-220.

[67] Khalifa, O.O., Densibali, A., and Faris, W., Image processing for chatter identification in machining processes, The International Journal of Advanced Manufacturing Technology, Vol. 31, No. 5-6, (2006), pp. 443-449.

[68] Szydłowski, M. and Powałka, B., Chatter detection algorithm based on machine vision, The International Journal of Advanced Manufacturing Technology, Vol. 62, No. 5-8, (2012), pp. 517-528.

[69] Szydłowski, M., Powałka, B., and Berczyński, S., Illumination for chatter marks detection using machine vision, Journal of Machine Engineering, Vol. 14, No.2, (2014), pp. 38-46.

[70] Lei, N. and Soshi, M., Vision-based system for chatter identification and process optimization in high-speed milling, The International Journal of Advanced Manufacturing Technology, Vol. 89, No. 9-12, (2017), pp. 2757-2769.

[71] Jeyapoovan, T. and Murugan, M., Surface roughness classification using image processing, Measurement, Vol. 46, No. 7, (2013), pp. 2065-2072.

[72] Tootooni, M.S., Liu, C., Roberson, D., Donovan, R., Rao, P.K., Kong, Z.J., and Bukkapatnam, S.T., Online non-contact surface finish measurement in machining

using graph theory-based image analysis, Journal of Manufacturing Systems, Vol. 41, (2016), pp. 266-276.

[73] Dhanasekar, B., Mohan, N.K., Bhaduri, B., and Ramamoorthy, B., Evaluation of surface roughness based on monochromatic speckle correlation using image processing, Precision Engineering, Vol. 32, No. 3, (2008), pp. 196-206.

[74] Lu, E., Liu, J., Gao, R., Yi, H., Wang, W., and Suo, X., Designing indices to measure surface roughness based on the color distribution statistical matrix (CDSM), Tribology International, Vol. 122, (2018), pp. 96-107.

[75] Suhail, S. M., Ali, J. M., Jailani, H. S., and Murugan, M., Vision based system for surface roughness characterisation of milled surfaces using speckle line images, In IOP Conference Series: Materials Science and Engineering, Vol. 402, No. 012054, (2018), pp. 1-12.

[76] Pour, M., Determining surface roughness of machining process types using a hybrid algorithm based on time series analysis and wavelet transform, The International Journal of Advanced Manufacturing Technology, Vol. 97, No. 5-8, (2018), pp. 2603-2619.

[77] Cireşan, D.C., Meier, U., Gambardella, L.M., and Schmidhuber, J., Deep, big, simple neural nets for handwritten digit recognition, Neural Computation, Vol. 22, No. 12, (2010), pp. 3207-3220.

[78] Eitel, A., Springenberg, J.T., Spinello, L., Riedmiller, M., and Burgard, W., Multimodal deep learning for robust RGB-D object recognition, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (2015), pp. 681-687.

[79] LeCun, Y., Kavukcuoglu, K., and Farabet, C., Convolutional networks and applications in vision, Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), (2010), pp. 253-256.

[80] Mordvintsev, A., Olah, C., and Tyka, M., Deepdream-a code example for visualizing neural networks, Google Research, Vol. 2, No. 5, (2015).

[81] Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B., and Liang, J., Convolutional neural networks for medical image analysis: Full training or fine tuning?, IEEE Transactions on Medical Imaging, Vol. 35, No. 5, (2016), pp. 1299-1312.

[82] Krizhevsky, A., Sutskever, I., and Hinton, G.E., Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, (2012), pp. 1097-1105.

[83] Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition, arXiv, (2014), 1409.1556.

[84] Vedaldi, A. and Lenc, K., Matconvnet: Convolutional neural networks for Matlab, Proceedings of the 23rd ACM International Conference on Multimedia, (2015), pp. 689-692.

[85] Li, Y., Wang, S., Tian, Q., and Ding, X., A survey of recent advances in visual feature detection, Neurocomputing, Vol. 149, (2015), No. 736-751.

[86] Davies, E.R., Computer and machine vision: theory, algorithms, practicalities, Academic Press, (2012).

[87] Harris, C. and Stephens, M.J., A combined corner and edge detector, Proceedings of the 4th Alvey Vision Conference, Vol. 15, No. 50, (1988), pp. 147–152.

[88] Shi, J. and Tomasi, C., Good features to track, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (1994), pp. 593–600.

[89] Rosten, E. and Drummond, T., Machine learning for high-speed corner detection, 9th European Conference on Computer Vision, Vol. 1, (2006), pp. 430–443.

[90] Leutenegger, S., Chli, M., and Siegwart, R., BRISK: Binary Robust Invariant Scalable Keypoints, Proceedings of the IEEE International Conference in Computer Vision (ICCV), (2011), 2548-2555.

[91] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L., Speeded-up robust features (SURF), Computer Vision and Image Understanding, Vol. 110, No. 3, (2008), pp. 346-359.

[92] Alahi, A., Ortiz, R., and Vandergheynst, P., Freak: Fast retina keypoint, IEEE Conference on Computer Vision and Pattern Recognition, (2012), pp. 510-517.

[93] Szeliski, R., Computer vision: algorithms and applications, Springer Science & Business Media, (2010).

[94] Fawcett, T., An introduction to ROC analysis, Pattern recognition letters, Vol. 27, No. 8, (2006), pp. 861-874.

[95] Torr, P.H. and Zisserman, A., MLESAC: A new robust estimator with application to estimating image geometry, Computer Vision and Image Understanding, Vol. 78, No. 1, (2000), pp. 138-156.

[96] Ren, S., Girshick, K.H.R., and Sun J., Faster R-CNN: Towards real-time object detection with region proposal networks, arXiv, (2015), 1506.01497.

[97] Canny, J., A computational approach to edge detection, IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, (1986), pp. 679–698.

[98] Sobel, I., An isotropic 3×3 gradient operator, machine vision for three-dimensional scenes, Freeman, H., Academic Press, New York, (1990), pp. 376-379.

[99] Ko, T.J., Park, J.W., Kim, H.S., and Kim, S.H., On-machine measurement using a non-contact sensor based on a CAD model, The International Journal of Advanced Manufacturing Technology, Vol. 32, No. 7-8, (2007), pp. 739-746.

[100] Krehel, R. and Pollák, M., The contactless measuring of the dimensional attrition of the cutting tool and roughness of machined surface, The International Journal of Advanced Manufacturing Technology, Vol. 86, No. 1-4, (2016), pp. 437-449.

[101] Smid, P., CNC control setup for milling and turning: mastering CNC control systems, Industrial Press Inc., (2010).

[102] Chen, Y.L., Cai, Y., Shimizu, Y., Ito, S., Gao, W., and Ju, B.F., On-machine measurement of microtool wear and cutting edge chipping by using a diamond edge artifact, Precision Engineering, Vol. 43, (2016), pp. 462-467.

[103] Lee, S.J., Kim, S.H., and Kim, O.H., The OMM System for machined form and surface roughness measurement concerned with volumetric error, Journal-Korean Society of Precision Engineering, 17(7), (2000), pp. 232-240.

[104] Ihara, Y. and Nagasawa, T., Fundamental study of the on-machine measurement in the machining center with a touch trigger probe, International Journal of Automation Technology, Vol. 7, No. 5, (2013), pp. 523-536.

[105] Gadelmawla, E.S., Koura, M.M., Maksoud, T.M.A., Elewa, I.M., and Soliman, H.H., Roughness parameters, Journal of Materials Processing Technology, Vol. 123, No. 1, (2002), pp. 133-145.

[106] Quintana, G. and Ciurana, J., Chatter in machining processes: A review, International Journal of Machine Tools and Manufacture, Vol. 51, No. 5, (2011), pp. 363-376.

[107] Siddhpura, M. and Paurobally, R., A review of chatter vibration research in turning, International Journal of Machine Tools and Manufacture, Vol. 61, (2012), pp. 27-47.

[108] Paris, S., Hasinoff, S.W., and Kautz, J., Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid, ACM Transactions on Graphics, Vol. 30(4), No. 68, (2011), pp. 1-12.

[109] Aubry, M., Paris, S., Hasinoff, S.W., Kautz, J., and Durand, F., Fast local laplacian filters: Theory and applications, ACM Transactions on Graphics (TOG), Vol. 33(5), No. 167, (2014), pp. 1-13.

[110] Pontius, R.G., Thontteh, O., and Chen, H., Components of information for multiple resolution comparison between maps that share a real variable, Environmental and Ecological Statistics, Vol. 15, No. 2, (2008), pp. 111-142.

APPENDIX

Appendix A.1 Cutting conditions and the result of measurement (using profilometer) of turned surfaces

Machine	**Material**	**ID**	**Setting value**			***Ra* [µm]**	***Rz* [µm]**	***RzJ* [µm]**	**Unstable cutting**
			***C* [m/min]**	***F* [mm/rev]**	***D* [mm]**				
Mazak Integrex I-100	50 mm ⌀ S45C	T1S	100	0.20	1.50	8.87	32.93	33.91	
		T2S	100	0.25	2.00	11.21	38.18	52.73	
		T3S	125	0.10	0.50	1.82	9.90	15.29	
		T4S	125	0.15	1.00	4.22	18.60	27.57	
		T5S	125	0.20	1.50	8.45	29.91	36.65	
		T6S	125	0.25	2.00	11.30	37.85	52.64	
		T7S	150	0.10	0.50	2.15	11.60	22.31	
		T8S	150	0.15	1.00	5.48	22.74	27.22	
		T9S	150	0.20	1.50	7.62	31.06	37.84	
		T10S	150	0.25	2.00	13.33	45.89	45.12	
		T11S	175	0.10	0.50	2.37	12.39	13.25	
		T12S	175	0.15	1.00	5.93	24.52	20.62	
		T13S	175	0.20	1.50	8.85	32.61	35.05	
		T14S	175	0.25	2.00	12.46	48.61	42.82	
		T15S	100	0.10	0.50	2.39	14.36	14.38	
		T16S	100	0.15	1.00	5.16	24.29	31.27	
		T17S	190	0.10	0.20	1.74	8.86	10.72	
		T18S	190	0.15	0.60	5.23	24.64	28.60	
		T19S	190	0.20	1.00	9.06	33.30	37.65	
		T20S	190	0.25	1.40	10.58	38.46	43.86	
		T21S	215	0.10	0.60	1.87	9.78	13.30	
		T22S	215	0.15	0.20	4.20	19.04	22.45	
		T23S	215	0.20	1.40	9.28	33.98	41.60	
		T24S	215	0.25	1.00	12.14	40.50	44.79	
		T25S	230	0.10	1.00	1.91	10.60	16.41	
		T26S	230	0.15	1.40	5.71	23.62	27.53	
		T27S	230	0.20	0.20	9.04	37.61	41.97	
		T28S	230	0.25	0.60	12.71	44.73	50.78	
		T29S	245	0.25	0.20	13.97	56.42	64.34	
		T30S	245	0.20	0.60	8.77	35.97	40.94	
		T31S	245	0.15	1.00	5.28	24.10	27.58	
		T32S	245	0.10	1.40	2.04	11.11	14.85	
Mazak Integrex I-100	40 mm ⌀ A5052	T33A	260	0.10	1.40	5.58	23.12	33.97	Yes
		T34A	260	0.15	1.00	4.87	19.45	22.72	
		T35A	260	0.20	0.60	7.24	26.19	29.26	
		T36A	260	0.25	0.20	11.44	40.92	47.21	
		T37A	260	0.10	1.40	11.75	46.11	57.33	Yes
		T38A	260	0.15	1.00	4.85	21.84	25.66	
		T39A	260	0.20	0.60	7.05	26.20	29.31	
		T40A	260	0.25	0.20	11.60	40.70	45.10	
		T41A	300	0.10	1.40	2.70	12.08	14.39	
		T42A	300	0.15	1.00	4.98	20.89	23.43	
		T43A	300	0.20	0.60	6.63	24.05	28.78	
		T44A	300	0.25	0.20	11.51	40.73	45.33	

Machine	Material	ID	Setting value			*Ra* [µm]	*Rz* [µm]	*RzJ* [µm]	Unstable
			C [m/min]	*F* [mm/rev]	*D* [mm]				cutting
		T45A	400	0.10	1.40	2.73	13.46	17.41	
		T46A	400	0.15	1.00	5.28	20.18	22.78	
		T47A	400	0.20	0.60	7.88	27.91	31.09	
		T48A	400	0.25	0.20	11.93	42.72	47.65	
Mazak Integrex I-100	30 mm ⌀ A5052	T49A	150	0.15	0.60	4.35	19.97	23.57	
		T50A	150	0.15	0.20	4.30	21.16	25.83	
		T51A	200	0.15	1.00	4.56	18.14	21.75	
		T52A	200	0.15	0.60	4.14	17.20	19.98	
		T53A	200	0.15	0.20	4.27	19.71	22.48	
		T54A	200	0.15	1.40	5.73	24.97	87.07	Yes
		T55A	250	0.15	0.60	4.45	20.33	23.98	
		T56A	250	0.15	0.20	4.09	20.61	24.61	
		T57A	250	0.15	1.40	4.79	19.76	23.74	
		T58A	300	0.15	1.40	34.06	159.49	268.32	Yes
		T59A	300	0.25	0.60	10.30	38.03	72.40	Yes
		T60A	350	0.10	1.40	17.68	67.57	93.51	Yes
		T61A	350	0.15	1.00	17.06	63.98	104.31	Yes
		T62A	350	0.20	0.60	24.39	89.52	156.10	Yes
		T63A	350	0.25	0.20	22.35	86.38	186.16	Yes
		T64A	100	0.10	1.00	1.88	9.94	12.75	
		T65A	100	0.15	0.60	4.39	21.43	28.12	
		T66A	100	0.20	0.20	6.71	32.70	40.30	
		T67A	125	0.10	1.00	1.81	11.07	14.61	
		T68A	125	0.15	0.60	4.07	21.61	28.43	
		T69A	125	0.20	0.20	6.37	32.24	42.36	
		T70A	175	0.20	1.00	7.01	29.37	47.27	
		T71A	175	0.15	0.60	4.04	19.16	22.66	
		T72A	175	0.10	0.20	1.39	6.61	8.24	
		T73A	225	0.10	1.00	1.74	10.80	14.53	
		T74A	225	0.15	0.60	4.20	18.66	21.50	
		T75A	225	0.20	0.20	7.22	30.40	33.27	
		T76A	275	0.10	1.00	3.97	18.98	23.51	Yes
		T77A	275	0.15	0.60	4.70	20.85	23.57	
		T78A	275	0.20	0.20	7.15	30.45	34.13	
		T79A	325	0.10	1.00	20.11	74.35	124.52	Yes
		T80A	325	0.15	0.60	4.48	19.10	22.68	
		T81A	325	0.20	0.20	7.38	31.06	35.30	
Mazak Integrex I-100	40 mm ⌀ A5052	T82A	100	0.10	1.40	1.74	8.56	10.78	
		T83A	100	0.15	1.00	3.29	16.66	21.66	
		T84A	100	0.20	0.60	6.10	25.32	29.94	
		T85A	100	0.25	0.20	9.51	44.36	52.73	
		T86A	150	0.10	1.40	1.83	9.28	11.05	
		T87A	150	0.15	1.00	3.91	18.31	22.46	
		T88A	150	0.20	0.60	6.19	24.41	28.16	
		T89A	150	0.25	0.20	11.24	46.05	50.20	
		T90A	200	0.10	1.40	8.26	32.01	39.65	Yes
		T91A	200	0.15	1.00	4.55	18.82	21.19	
		T92A	200	0.20	0.60	7.00	25.93	28.94	
		T93A	200	0.25	0.20	11.78	47.24	51.43	
		T94A	225	0.10	1.40	11.92	44.69	61.62	Yes
		T95A	225	0.15	1.00	4.74	20.49	23.19	
		T96A	225	0.20	0.60	6.82	27.66	31.09	
		T97A	225	0.25	0.20	11.80	49.12	53.40	

Machine	**Material**	**ID**	**Setting value**			***Ra* [µm]**	***Rz* [µm]**	***RzJ* [µm]**	**Unstable cutting**
			S [rpm]	***F* [mm/rev]**	***D* [mm]**				
Howa Strong 860	50 mm ⌀ S45C	T98S	380	0.10	0.20	2.72	15.71	20.92	
		T99S	380	0.15	0.60	2.26	12.65	15.78	
		T100S	380	0.20	1.00	3.16	11.95	14.33	
		T101S	380	0.25	1.40	3.91	18.09	23.41	
		T102S	600	0.10	0.60	1.26	8.30	10.83	
		T103S	600	0.15	0.20	2.30	12.12	14.75	
		T104S	600	0.20	1.40	3.27	12.97	17.47	
		T105S	600	0.25	1.00	3.88	16.79	19.92	
		T106S	950	0.10	1.00	1.26	7.48	10.25	
		T107S	950	0.15	1.40	2.36	12.02	18.43	
		T108S	950	0.20	0.20	3.11	11.83	18.08	
		T109S	950	0.25	0.60	3.78	17.13	22.84	Yes
		T110S	1500	0.10	1.40	1.73	10.64	15.05	
		T111S	1500	0.15	1.00	2.42	12.67	18.23	
		T112S	1500	0.20	0.60	3.26	13.46	19.51	
		T113S	1500	0.25	0.20	3.89	17.26	29.73	
		T114S	950	0.25	1.40	4.06	18.20	38.51	Yes
		T115S	950	0.25	1.00	3.88	16.77	35.25	Yes

Appendix A.2 Cutting conditions and the result of measurement (using profilometer) slot milled turned surfaces

Machine	**Tool**	**ID**	**Setting value**			***Ra* [µm]**	***Rz* [µm]**	***RzJ* [µm]**	**Unstable cutting**
			***S* [rpm]**	***F* [mm/min]**	***D* [mm]**				
Hitachi 2MW-V	10 mm ⌀ HSS end mill	L1S	455	50	0.50	5.10	24.26	33.84	
		L2S	455	100	1.00	5.29	26.26	35.79	
		L3S	455	150	1.50	7.04	33.68	43.11	
		L4S	455	200	2.00	8.05	38.53	54.77	
		L5S	610	50	1.00	3.20	18.31	27.56	
		L6S	610	100	0.50	4.85	22.91	29.16	
		L7S	610	150	2.00	5.77	26.89	36.08	
		L8S	610	200	1.50	6.66	32.21	41.41	
		L9S	1100	50	1.50	4.84	20.53	26.08	
		L10S	1100	100	2.00	4.03	21.61	33.98	
		L11S	1100	150	0.50	4.66	24.33	32.54	
		L12S	1100	200	1.00	3.88	21.24	34.44	
		L13S	1500	50	2.00	3.84	19.43	30.43	
		L14S	1500	100	1.50	3.04	14.88	18.82	
		L15S	1500	150	1.00	3.65	20.94	31.31	
		L16S	1500	200	0.50	4.50	22.43	29.84	
Hitachi 2MW-V	10 mm ⌀ HSS end mill	L17A	455	15	1.00	0.25	2.08	6.11	
		L18A	455	30	1.50	0.88	6.48	7.21	
		L19A	455	50	2.00	1.24	6.56	10.90	
		L20A	610	30	1.00	0.40	3.47	4.91	
		L21A	610	50	0.50	0.87	4.68	5.43	
		L22A	610	100	2.00	1.23	7.57	5.88	
		L23A	610	150	1.50	1.39	8.10	6.79	
		L24A	790	30	1.50	0.46	3.83	11.19	

Machine	Tool	ID	Setting value			*Ra* [µm]	*Rz* [µm]	*RzJ* [µm]	Unstable cutting
			S [rpm]	*F* [mm/min]	*D* [mm]				
		L25A	790	50	2.00	0.67	5.36	11.20	
		L26A	790	100	0.50	1.09	5.21	8.05	
		L27A	790	150	1.00	1.27	6.21	6.13	
		L28A	1100	50	1.50	0.42	3.41	9.19	
		L29A	1100	100	1.00	0.70	4.29	5.81	
		L30A	1100	150	0.50	1.13	5.18	5.15	
Hitachi	8 mm ⌀	L31A	455	50	1.00	0.68	4.94	7.71	
2MW-V	NS	L32A	455	100	0.50	1.45	9.16	8.35	
	MSE230	L33A	790	50	1.50	0.72	5.95	8.12	
	carbide	L34A	790	100	2.00	2.09	13.32	4.56	
	end mill	L35A	790	150	0.50	0.76	4.79	11.83	
	with	L36A	790	200	1.00	0.91	5.82	8.32	
	TiAlN	L37A	1100	100	1.50	0.92	6.79	7.44	
	coating	L38A	1100	150	1.00	0.68	4.80	8.13	
		L39A	1100	200	0.50	0.77	5.25	7.47	
		L40A	1500	150	1.00	0.72	5.23	5.85	
		L41A	1500	200	0.50	0.65	4.53	7.04	
Makino	10 mm ⌀	L42S	6500	400	0.50	0.86	4.58	7.51	Yes
Seiki	NS	L43S	6500	550	1.00	1.11	5.69	6.00	Yes
MSA30	MSE230	L44S	8000	200	2.00	1.38	7.47	6.23	Yes
	carbide	L45S	8000	280	1.50	0.45	3.00	7.35	
	end mill	L46S	8000	400	1.00	0.56	3.57	6.47	
	with	L47S	8000	550	0.50	0.67	3.93	4.83	Yes
	TiAlN	L48S	2800	280	2.00	0.56	3.74	5.11	
	coating	L49S	6000	550	2.00	1.57	7.71	6.43	Yes
		L50S	8000	550	2.00	1.33	7.26	6.40	Yes
		L51S	3500	200	0.50	0.82	5.18	5.14	Yes
		L52S	3500	280	1.00	0.50	3.16	3.92	Yes
		L53S	3500	400	1.50	0.86	4.92	4.87	Yes
		L54S	3500	550	2.00	0.49	3.18	5.86	
		L55S	5000	200	1.00	0.54	3.29	6.76	Yes
		L56S	5000	280	0.50	0.82	4.52	3.70	Yes
		L57S	5000	400	2.00	0.85	5.34	4.22	Yes
		L58S	5000	550	1.50	0.63	3.99	5.31	Yes
		L59S	6500	200	1.50	0.49	2.95	4.03	Yes
		L60S	6500	280	2.00	1.43	7.43	5.59	Yes
Makino	8 mm ⌀	L61S	2500	200	2.00	0.79	5.24	3.68	
Seiki	NS	L62S	2500	280	1.50	0.80	5.52	6.60	
MSA30	MSE230	L63S	2500	400	1.00	0.92	5.33	4.58	
	carbide	L64S	2500	550	0.50	1.16	6.11	5.85	
	end mill	L65S	3000	200	1.50	0.52	4.09	4.75	
	with	L66S	3000	280	1.00	0.64	5.11	3.48	
	TiAlN	L67S	3000	400	0.50	0.80	5.65	12.62	
	coating	L68S	3000	550	2.00	0.68	4.06	12.37	
		L69S	4000	200	1.00	0.51	4.05	5.53	Yes
		L70S	4000	280	0.50	0.77	5.37	6.54	Yes
		L71S	4000	400	2.00	0.71	4.35	14.89	Yes
		L72S	4000	550	1.50	0.61	3.46	14.60	
		L73S	4500	200	0.50	0.58	3.64	8.26	Yes
		L74S	4500	280	2.00	0.88	4.60	10.52	Yes
		L75S	4500	400	1.50	0.92	4.87	7.64	Yes
		L76S	4500	550	1.00	0.59	3.38	11.30	
		L77A	2000	200	1.00	0.43	2.54	6.74	

Machine	**Tool**	**ID**	**Setting value**			*Ra* [µm]	*Rz* [µm]	*RzJ* [µm]	**Unstable cutting**
			S [rpm]	*F* [mm/min]	*D* [mm]				
Makino Seiki MSA30	10 mm ⌀ NS MSE230 carbide end mill with TiAlN coating	L78A	2000	280	1.50	0.51	3.24	6.96	
		L79A	2000	400	2.00	0.71	4.19	8.03	
		L80A	3000	200	1.00	0.85	4.56	12.17	Yes
		L81A	3000	280	0.50	0.37	2.25	8.17	
		L82A	3000	400	2.00	0.52	2.90	12.60	
		L83A	3000	550	1.50	0.63	3.66	12.35	
		L84A	4000	200	1.50	0.50	2.69	8.21	Yes
		L85A	4000	280	2.00	0.74	4.04	7.42	Yes
		L86A	4000	400	0.50	0.39	2.21	9.11	
		L87A	4000	550	1.00	1.14	5.42	6.76	Yes
		L88A	5000	280	1.50	0.57	3.17	7.22	Yes
		L89A	5000	400	1.00	0.80	4.53	7.73	Yes
		L90A	5000	550	0.50	0.47	2.78	6.94	
Makino Seiki MSA30	8 mm ⌀ NS MSE230 carbide end mill with TiAlN coating	L91A	1500	200	1.00	0.62	4.65	9.33	
		L92A	1500	280	1.50	0.78	5.14	5.46	
		L93A	1500	400	2.00	1.16	6.34	8.46	
		L94A	2500	200	1.00	0.46	3.80	8.72	
		L95A	2500	280	0.50	0.55	3.91	11.45	
		L96A	2500	400	2.00	0.65	4.30	18.25	
		L97A	2500	550	1.50	0.89	5.33	11.93	
		L98A	3500	200	1.50	0.45	3.37	30.70	
		L99A	3500	280	2.00	0.46	3.41	6.88	
		L100A	3500	400	0.50	0.56	4.03	8.69	
		L101A	3500	550	1.00	0.66	4.28	15.31	
		L102A	4500	200	1.50	0.65	4.38	7.82	Yes
		L103A	4500	400	1.00	0.48	3.68	10.49	
		L104A	4500	280	2.00	0.71	4.63	10.35	Yes
		L105A	5500	400	1.50	0.83	5.44	8.48	Yes

Appendix A.3 Cutting conditions and the result of measurement (using profilometer) of side milled surfaces

Machine	**Tool**	**ID**	**Setting value**			*Ra* [µm]	*Rz* [µm]	*RzJ* [µm]	**Unstable cutting**
			S [rpm]	*F* [mm/min]	*D* [mm]				
Hitachi 2MW-V	10 mm ⌀ HSS end mill	E1S	610	50	0.40	0.99	6.29	11.40	
		E2S	610	100	0.60	1.23	7.26	16.41	
		E3S	610	150	0.80	2.56	12.02	23.21	Yes
		E4S	610	200	1.00	3.42	14.67	34.44	Yes
		E5S	790	50	0.60	1.15	6.74	11.03	
		E6S	790	100	0.40	1.34	7.35	11.79	
		E7S	790	150	1.00	0.88	5.36	11.45	Yes
		E8S	790	200	0.80	1.30	7.37	13.95	Yes
		E9S	1100	50	0.80	1.47	9.42	20.55	Yes
		E10S	1100	100	1.00	1.20	7.13	14.79	Yes
		E11S	1100	150	0.40	1.56	9.06	16.54	Yes
		E12S	1100	200	0.60	1.53	8.61	16.90	Yes
		E13S	1500	50	0.40	1.55	9.36	17.91	
		E14S	1500	100	0.60	1.70	10.11	17.27	Yes
		E15S	1500	150	0.80	1.68	10.19	20.08	Yes

Machine	Tool	ID	Setting value			*Ra* [µm]	*Rz* [µm]	*RzJ* [µm]	Unstable cutting
			S **[rpm]**	***F*** **[mm/min]**	***D*** **[mm]**				
		E16S	1500	200	1.00	2.28	12.20	32.46	Yes
		E17S	1500	200	0.40	1.81	10.04	16.10	
		E18S	1500	50	1.00	1.61	10.16	18.95	
Hitachi 2MW-V	10 mm ⌀ HSS end mill	E19A	610	100	0.80	0.36	2.54	7.08	
		E20A	610	150	0.60	0.40	2.48	6.32	
		E21A	610	200	0.40	0.45	2.75	5.95	
		E22A	790	100	1.00	0.33	2.30	6.45	
		E23A	790	150	0.80	0.34	2.12	5.66	
		E24A	790	200	0.60	0.38	2.40	5.12	
		E25A	790	250	0.40	0.47	3.09	6.49	
		E26A	1100	150	1.00	0.33	2.21	4.73	
		E27A	1100	200	0.80	0.35	2.36	5.14	
		E28A	1100	250	0.60	0.36	2.23	5.42	
		E29A	1100	300	0.40	0.40	2.45	4.83	
		E30A	1500	200	1.00	0.38	2.40	5.67	
		E31A	1500	250	0.80	0.37	2.27	6.10	
		E32A	1500	300	0.60	0.38	2.53	5.56	
		E33A	1500	350	0.40	0.37	2.61	6.05	
Makino Seiki MSA30	10 mm ⌀ NS MSE230 carbide end mill with TiAlN coating	E34S	2500	200	0.40	0.68	4.08	4.69	
		E35S	2500	200	0.60	0.64	3.71	7.15	
		E36S	2500	200	0.80	0.59	3.50	5.48	
		E37S	2500	200	1.00	0.64	3.75	9.74	
		E38S	5000	400	0.40	0.78	3.95	8.28	
		E39S	5000	400	0.60	0.45	2.76	4.89	
		E40S	5000	400	0.80	0.42	2.73	5.27	
		E41S	5000	400	1.00	0.40	2.42	5.87	
		E42S	6875	550	0.40	0.76	3.86	5.19	
		E43S	6875	550	0.60	0.64	3.38	6.38	
		E44S	6875	550	0.80	0.53	2.84	6.20	
		E45S	6875	550	1.00	0.63	3.37	7.95	Yes
		E46S	9375	750	0.40	0.65	3.56	6.12	
		E47S	9375	750	0.60	0.64	3.24	8.42	
		E48S	9375	750	0.80	0.57	3.04	17.02	Yes
		E49S	9375	750	1.00	0.48	2.71	9.85	Yes
		E50S	2000	200	1.00	0.71	4.27	7.49	
		E51S	2000	280	0.80	0.81	5.04	7.16	
		E52S	2000	400	0.60	0.80	5.35	9.37	
		E53S	2000	550	0.40	0.84	5.50	12.43	
		E54S	3000	280	1.00	0.70	4.68	9.33	Yes
		E55S	3000	400	0.80	0.63	4.03	9.51	Yes
		E56S	3000	550	0.60	0.64	4.06	6.72	Yes
		E57S	3000	750	0.40	0.67	4.21	7.47	
		E58S	4000	280	1.00	0.53	3.64	6.84	Yes
		E59S	4000	400	0.80	0.46	2.91	7.18	Yes
		E60S	4000	550	0.60	0.52	3.44	5.58	Yes
		E61S	4000	750	0.40	0.57	3.69	6.30	
		E62S	5000	280	1.00	0.56	3.54	6.71	Yes
		E63S	5000	400	0.80	0.49	3.11	6.24	Yes
		E64S	5000	550	0.60	0.50	3.38	6.19	Yes
		E65S	5000	750	0.40	0.40	2.70	7.08	
Makino Seiki MSA30	8 mm ⌀ NS	E66S	1500	280	1.00	0.57	3.64	5.07	
		E67S	1500	400	0.80	0.72	4.41	7.60	
		E68S	1500	550	0.60	0.75	4.64	5.27	

Machine	Tool	ID	Setting value			*Ra* [µm]	*Rz* [µm]	*RzJ* [µm]	Unstable cutting
			S **[rpm]**	***F*** **[mm/min]**	***D*** **[mm]**				
	MSE230	E69S	1500	750	0.40	0.98	5.95	5.98	
	carbide	E70S	2500	280	1.00	1.07	6.28	5.47	Yes
	end mill	E71S	2500	400	0.80	0.89	5.47	4.40	Yes
	with	E72S	2500	550	0.60	0.69	4.23	5.34	
	TiAlN	E73S	2500	750	0.40	0.87	5.23	7.02	
	coating	E74S	3500	280	1.00	1.15	6.37	5.47	Yes
		E75S	3500	400	0.80	0.54	3.78	6.65	
		E76S	3500	550	0.60	0.65	4.06	4.41	
		E77S	3500	750	0.40	0.59	3.64	4.99	
		E78S	4500	280	1.00	1.99	11.09	5.25	Yes
		E79S	4500	400	0.80	1.83	10.41	5.04	Yes
		E80S	4500	550	0.60	1.00	5.51	6.96	Yes
		E81S	4500	750	0.40	1.14	5.67	6.78	
Makino	8 mm ⌀	E82A	1500	200	1.00	0.43	2.98	6.93	
Seiki	NS	E83A	1500	280	0.80	0.41	2.59	7.50	
MSA30	MSE230	E84A	1500	400	0.60	0.57	3.67	9.09	
	carbide	E85A	1500	550	0.40	0.72	4.16	10.05	
	end mill	E86A	2500	200	1.00	0.41	2.63	11.18	
	with	E87A	2500	280	0.80	0.44	2.77	11.61	
	TiAlN	E88A	2500	400	0.60	0.41	2.72	6.94	
	coating	E89A	2500	550	0.40	0.38	2.54	10.69	
		E90A	3500	200	1.00	0.62	3.89	15.92	Yes
		E91A	3500	280	0.80	0.58	3.60	10.64	Yes
		E92A	3500	400	0.60	0.60	3.94	9.35	Yes
		E93A	3500	550	0.40	0.56	3.51	7.28	
		E94A	4500	200	1.00	0.78	4.94	20.07	Yes
		E95A	4500	280	0.80	1.10	6.53	19.47	Yes
		E96A	4500	400	0.60	0.98	5.70	8.53	Yes
		E97A	4500	550	0.40	0.70	4.09	9.72	Yes
Makino	10 mm ⌀	E98A	1000	280	1.00	0.52	2.83	6.49	
Seiki	NS	E99A	1000	400	0.80	1.04	4.82	8.40	
MSA30	MSE230	E100A	1000	550	0.60	1.64	7.81	13.37	
	carbide	E101A	1000	750	0.40	1.94	8.02	17.33	
	end mill	E102A	2000	280	1.00	0.45	2.47	5.18	
	with	E103A	2000	400	0.80	0.40	2.30	4.80	
	TiAlN	E104A	2000	550	0.60	0.55	3.00	5.19	
	coating	E105A	2000	750	0.40	0.95	4.53	7.50	
		E106A	3000	280	1.00	1.43	7.29	11.02	Yes
		E107A	3000	400	1.80	2.50	11.92	21.24	Yes
		E108A	3000	550	1.60	1.03	4.78	9.65	Yes
		E109A	3000	750	0.40	0.61	3.47	6.29	
		E110A	4000	280	1.00	1.42	7.50	10.95	Yes
		E111A	4000	400	0.80	1.15	5.84	9.67	Yes
		E112A	4000	550	0.60	0.46	2.61	4.85	
		E113A	4000	750	0.40	0.52	2.79	5.74	

Appendix B.1 The complete results of the training and prediction process using the five loss functions for turned surfaces

Loss function	**Case**	**RMSE**	***R***	***R*^{2}**	**Adjusted - *R*^{2}**	**Average accuracy**	**Training time [h]**	**Test time /sample [s]**
MSE	cross	0.49	1.00	0.99	0.99	93.53%	24.34	0.37
	1	1.29	0.97	0.94	0.94	89.95%	25.63	0.38
	2	1.18	0.98	0.97	0.97	91.24%	25.76	0.37
	3	2.40	0.95	0.90	0.90	87.44%	26.16	0.37
MAE	cross	0.49	1.00	0.99	0.99	95.09%	26.53	0.37
	1	1.55	0.96	0.92	0.92	91.08%	18.51	0.36
	2	1.22	0.98	0.97	0.97	93.52%	18.06	0.36
	3	2.40	0.93	0.87	0.86	86.35%	24.67	0.36
MAPE	cross	0.56	0.99	0.99	0.99	94.69%	19.30	0.37
	1	1.37	0.97	0.94	0.94	91.67%	24.48	0.37
	2	1.26	0.98	0.97	0.97	93.15%	26.93	0.38
	3	1.80	0.96	0.91	0.91	85.83%	17.48	0.36
Log-cosh	cross	0.42	1.00	0.99	0.99	94.51%	27.11	0.36
	1	1.34	0.97	0.94	0.94	91.07%	23.69	0.37
	2	1.24	0.98	0.96	0.96	92.77%	14.98	0.36
	3	2.20	0.95	0.89	0.89	87.35%	25.42	0.36
Huber (4.0)	cross	0.41	1.00	0.99	0.99	94.39%	31.42	0.36
	1	1.56	0.97	0.93	0.93	90.21%	21.66	0.36
	2	0.99	0.99	0.97	0.97	92.39%	28.93	0.36
	3	3.25	0.93	0.86	0.86	83.44%	21.70	0.36

Appendix B.2 The complete results of the training and prediction process using the five loss functions for slot milled surfaces

Loss function	**Case**	**RMSE**	***R***	***R*^{2}**	**Adjusted - *R*^{2}**	**Average accuracy**	**Training time [h]**	**Test time /sample [s]**
MSE	cross	0.29	0.99	0.97	0.97	87.95%	13.74	0.23
	1	0.29	1.00	0.99	0.99	88.98%	14.85	0.23
	2	0.49	0.95	0.91	0.90	81.73%	15.15	0.22
	3	0.22	0.99	0.97	0.97	87.64%	12.83	0.22
MAE	cross	0.26	0.99	0.98	0.98	92.17%	6.92	0.23
	1	0.46	0.99	0.99	0.99	87.65%	13.77	0.23
	2	0.44	0.96	0.91	0.91	84.45%	7.90	0.23
	3	0.29	0.98	0.95	0.95	84.78%	8.48	0.23
MAPE	cross	0.36	0.98	0.95	0.95	93.71%	10.12	0.23
	1	0.37	0.99	0.99	0.99	88.93%	13.13	0.23
	2	0.44	0.96	0.92	0.92	85.07%	11.15	0.23
	3	0.20	0.99	0.98	0.98	86.34%	11.68	0.23
Log-cosh	cross	0.29	0.98	0.97	0.97	88.54%	6.74	0.23
	1	0.41	1.00	0.99	0.99	88.79%	6.75	0.23
	2	0.46	0.96	0.92	0.92	86.06%	6.80	0.23
	3	0.24	0.98	0.97	0.97	85.80%	8.44	0.22
Huber (1.5)	cross	0.25	0.99	0.98	0.98	88.73%	7.84	0.23
	1	0.40	1.00	0.99	0.99	85.41%	8.50	0.22
	2	0.47	0.96	0.92	0.92	82.64%	8.86	0.22
	3	0.19	0.99	0.98	0.98	86.83%	8.16	0.23

Appendix B.3 The complete results of the training and prediction process using the five loss functions for side milled surfaces

Loss function	**Case**	**RMSE**	***R***	***R*^{2}**	**Adjusted - *R*^{2}**	**Average accuracy**	**Training time [h]**	**Test time /sample [s]**
MSE	cross	0.08	0.99	0.98	0.98	92.11%	10.42	0.22
	1	0.14	0.97	0.94	0.94	87.75%	4.51	0.22
	2	0.19	0.97	0.94	0.94	87.80%	7.33	0.22
	3	0.12	0.96	0.93	0.93	87.46%	7.67	0.22
MAE	cross	0.09	0.99	0.98	0.98	93.13%	8.82	0.22
	1	0.12	0.98	0.96	0.96	90.68%	4.81	0.23
	2	0.18	0.98	0.96	0.96	89.29%	12.66	0.23
	3	0.10	0.97	0.93	0.93	88.41%	7.19	0.22
MAPE	cross	0.09	0.99	0.97	0.97	91.18%	9.14	0.22
	1	0.12	0.98	0.96	0.95	90.53%	9.97	0.22
	2	0.19	0.97	0.94	0.94	88.20%	9.29	0.22
	3	0.12	0.97	0.94	0.94	86.22%	7.63	0.26
Log-cosh	cross	0.08	0.99	0.98	0.98	93.99%	7.44	0.22
	1	0.13	0.98	0.96	0.96	92.03%	10.33	0.22
	2	0.18	0.98	0.95	0.95	90.73%	9.28	0.23
	3	0.11	0.97	0.93	0.93	88.23%	8.16	0.23
Huber (0.5)	cross	0.08	0.99	0.98	0.98	91.80%	7.06	0.22
	1	0.13	0.97	0.94	0.94	88.94%	6.46	0.22
	2	0.21	0.96	0.93	0.92	87.87%	8.55	0.25
	3	0.09	0.97	0.95	0.95	88.60%	9.11	0.23

Appendix C.1 Results of the combined *Ra*-unstable cutting evaluation architecture for turned surfaces

Architecture	**Case**	***R*^{2}**	**Adjusted - *R*^{2}**	***Ra* accuracy**	**Chatter accuracy**	**Training time [h]**	**Test time /sample [s]**
1	cross	0.99	0.99	94.29%	100%	32.67	0.64
	1	0.95	0.94	89.79%	93%	33.36	0.61
	2	0.96	0.96	92.00%	100%	32.80	0.62
	3	0.89	0.89	86.85%	98%	35.60	0.63
2	cross	0.99	0.99	93.93%	100%	27.19	0.38
	1	0.94	0.94	89.50%	91%	28.38	0.38
	2	0.97	0.97	92.14%	94%	28.79	0.39
	3	0.89	0.89	87.06%	100%	33.60	0.39
3	cross	1.00	1.00	95.53%	100%	56.51	3.07
	1	0.93	0.93	88.96%	95%	63.11	3.06
	2	0.97	0.97	93.72%	97%	57.58	3.07
	3	0.91	0.91	86.28%	100%	61.11	3.07
4	cross	1.00	1.00	96.07%	100%	57.01	1.92
	1	0.93	0.93	87.87%	99%	53.26	1.91
	2	0.97	0.97	93.32%	98%	59.88	1.90
	3	0.90	0.90	87.35%	100%	58.75	1.93

Appendix C.2 Results of the combined *Ra*-unstable cutting evaluation architecture for slot milled surfaces

Architecture	**Case**	***R*²**	**Adjusted - *R*²**	***Ra* accuracy**	**Chatter accuracy**	**Training time [h]**	**Test time /sample [s]**
1	cross	0.97	0.97	87.81%	100%	20.30	0.39
	1	0.99	0.99	87.59%	95%	20.84	0.39
	2	0.90	0.90	83.19%	90%	20.98	0.38
	3	0.96	0.96	84.56%	90%	24.08	0.38
2	cross	0.97	0.97	88.58%	100%	15.83	0.22
	1	0.99	0.99	87.78%	92%	14.64	0.23
	2	0.90	0.90	84.36%	94%	14.25	0.22
	3	0.95	0.95	83.66%	90%	15.95	0.23
3	cross	0.99	0.98	93.36%	100%	48.44	1.84
	1	0.99	0.99	86.91%	93%	44.60	1.91
	2	0.92	0.92	82.47%	89%	49.56	1.92
	3	0.98	0.98	83.95%	96%	43.95	1.91
4	cross	0.99	0.99	93.69%	100%	38.48	1.06
	1	0.99	0.99	86.21%	98%	39.95	1.03
	2	0.92	0.92	82.39%	97%	33.58	1.03
	3	0.98	0.98	86.28%	100%	37.77	1.03

Appendix C.3 Results of the combined *Ra*-unstable cutting evaluation architecture for side milled surfaces

Architecture	**Case**	***R*²**	**Adjusted - *R*²**	***Ra* accuracy**	**Chatter accuracy**	**Training time [h]**	**Test time /sample [s]**
1	cross	0.97	0.97	93.49%	99%	16.12	0.38
	1	0.96	0.96	90.76%	90%	17.17	0.39
	2	0.87	0.87	88.05%	72%	13.89	0.39
	3	0.95	0.95	89.64%	90%	22.20	0.38
2	cross	0.98	0.98	94.09%	97%	12.00	0.23
	1	0.96	0.95	90.14%	93%	11.63	0.23
	2	0.94	0.94	91.08%	66%	12.44	0.23
	3	0.96	0.96	89.76%	81%	15.03	0.22
3	cross	0.99	0.99	94.88%	98%	34.84	1.95
	1	0.98	0.98	91.37%	100%	32.05	1.92
	2	0.95	0.95	91.01%	75%	49.03	1.93
	3	0.97	0.97	89.57%	90%	47.61	1.94
4	cross	0.98	0.98	95.15%	98%	19.15	1.25
	1	0.97	0.97	92.07%	99%	28.42	1.18
	2	0.92	0.92	91.36%	75%	25.17	1.05
	3	0.97	0.97	90.74%	90%	17.55	1.04

Appendix D Results of the combined *Ra*-unstable cutting evaluation models with MSE and MAPE loss

Datasets	**Loss functions**	**Case**	***R^2***	**Adjusted - *R^2***	***Ra* accuracy**	**Chatter accuracy**	**Training time [h]**	**Test time /sample [s]**
Turned surface	MSE	cross	0.99	0.99	93.93%	100%	27.19	0.38
		1	0.94	0.94	89.50%	91%	28.38	0.38
		2	0.97	0.97	92.14%	94%	28.79	0.39
		3	0.89	0.89	87.06%	100%	33.60	0.39
	MAPE	cross	0.99	0.99	94.75%	100%	30.56	0.37
		1	0.94	0.94	90.58%	98%	30.63	0.37
		2	0.98	0.98	93.88%	100%	18.98	0.37
		3	0.92	0.92	87.57%	100%	30.30	0.38
Slot milled surface	MSE	cross	0.97	0.97	88.58%	100%	15.83	0.22
		1	0.99	0.99	87.78%	92%	14.64	0.23
		2	0.90	0.90	84.36%	94%	14.25	0.22
		3	0.95	0.95	83.66%	90%	15.95	0.23
	MAPE	cross	0.96	0.95	93.59%	100%	14.83	0.23
		1	0.99	0.99	90.49%	96%	15.14	0.23
		2	0.91	0.91	87.49%	93%	12.51	0.23
		3	0.98	0.98	86.69%	90%	11.49	0.24
Side milled surface	MSE	cross	0.97	0.97	89.76%	99%	6.66	0.23
		1	0.95	0.94	85.38%	100%	7.96	0.23
		2	0.94	0.94	87.59%	72%	8.43	0.22
		3	0.94	0.94	85.23%	82%	10.52	0.22
	MAPE	cross	0.98	0.98	94.09%	97%	12.00	0.23
		1	0.96	0.95	90.14%	93%	11.63	0.23
		2	0.94	0.94	91.08%	66%	12.44	0.23
		3	0.96	0.96	89.76%	81%	15.03	0.22

LIST OF ACHIEVEMENTS

Articles on periodicals (related to thesis):

1) **Achmad P. Rifai**, Ryo Fukuda, and Hideki Aoyama, "Image Based Identification of Cutting Tools in Turning-Milling Machines", Journal of the Japan Society for Precision Engineering, Vol. 85, No. 2, (2019), pp. 159-166.

2) **Achmad P. Rifai**, Ryo Fukuda, and Hideki Aoyama, "Surfaces Roughness Estimation and Chatter Vibration Identification Using Vision-Based Deep Learning", Journal of the Japan Society for Precision Engineering, Vol. 85, No. 7, (2019), pp. 658-666.

Articles on periodicals:

1) **Achmad P. Rifai**, Siti Z. M. Dawal, Aliq Zuhdi, Hideki Aoyama, and Keith Case, "Reentrant FMS scheduling in loop layout with consideration of multi loading-unloading stations and shortcuts", The International Journal of Advanced Manufacturing Technology, Vol. 82, (2016), pp. 1527–1545.

2) Nguyen Huu Tho, Siti Z. M. Dawal, Nukman Yusoff, **Achmad P. Rifai**, and Hideki Aoyama, "An integrated MCDM model for conveyor equipment evaluation and selection in an FMC based on a fuzzy AHP and fuzzy ARAS in the presence of vagueness", PloS one, Vol. 11, No. 4, (2016), e0153222.

3) **Achmad P. Rifai**, Nguyen Huu Tho, and Siti Z. M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling", Applied Soft Computing, Vol. 40, (2016), pp. 42-57.

4) **Achmad P. Rifai**, Nguyen Huu Tho, Hideki Aoyama, Siti Z. M. Dawal, and Nur Aini Masruroh, "Non-dominated sorting biogeography-based optimization for bi-objective reentrant flexible manufacturing system scheduling", Applied Soft Computing, Vol. 62, (2018), pp. 187-202.

Articles on international conference proceedings:

1) **Achmad P. Rifai,** Hideki Aoyama, "Identification of Cutting Tool for Avoiding Collision in Turning-Milling Machine", Proceedings of The 20th International Symposium on Advances in Abrasive Technology, Okinawa, Japan, 3-6 December 2017, pp. 477-482.

2) **Achmad P. Rifai**, Ryo Fukuda, and Hideki Aoyama, “Identification of Cutting Tool for Avoiding Collision in Turning-Milling Machine”, Proceedings of The 17th International Conference on Precision Engineering, Kamakura, Japan, 12-16 November 2018.

Presentations at international conferences:

Not applicable

Presentations at domestic meetings:

Not applicable

Others:

Not applicable