Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 48 (2020) 947–958

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

48th SME North American Manufacturing Research Conference, NAMRC 48 (Cancelled due to COVID-19)

Digital image processing with deep learning for automated cutting tool wear detection

Thomas Bergs[ab], Carsten Holst[b*], Pranjul Gupta[b], Thorsten Augspurger[a]

[a]*Werkzeugmaschinenlabor WZL der RWTH Aachen, Campus-Boulevard 30, 52074 Aachen, Germany*
[b]*Fraunhofer Institute for Production Technology, Steinbachstraße 17, 52074 Aachen, Germany*

* Carsten Holst. Tel.: +49-241-8904-123 ; Fax: +49-241-8904-6123. *E-mail address:* carsten.holst@ipt.fraunhofer.de

Abstract

Tool wear is a cost driver in the metal cutting industry. Besides costs for the cutting tools themselves, further costs appear - equipment downtime for tool changes, reworking of damaged surfaces, scrap parts or damages to the machine tool itself in the worst case. Consequently, tools need to be exchanged on a regular basis or at a defined tool wear state. In order to detect and monitor the tool wear state different approaches are possible. In this publication, a deep learning approach for image processing is investigated in order to quantify the tool wear state. In a first step, a Convolutional Neural Networks (CNN) is trained for cutting tool type classification. This works well with an accuracy of 95.6% on the test dataset. Finally, a Fully Convolutional Network (FCN) for semantic segmentation is trained on individual tool type datasets (ball end mill, end mill, drills and inserts) and a mixed dataset to detect worn areas on the microscopic tool images. The accuracy metric for this kind of task, Intersect over Union (IoU), is around 0.7 for all networks on the test dataset. This paper contributes to the perspective of a fully automated cutting tool wear analysis method using machine tool integrated microscopes in the scientific and industrial environment.

Keywords: Tool Wear Detection; Cutting Tools; Semantic Segmentation; Deep Learning; Convolutional Neural Networks; Fully Convolutional Networks

1. Introduction

In machining proceeding, wear of the cutting tool causes a continuous change of the process state variables in the cutting zone like forces and temperatures acting on the workpiece and tool. Those process conditions on their part influence the tool wear rate itself as well as the process induced surface modifications of the workpiece and finally the geometrical accuracy of the manufactured part. The tool wear rate depends on the process parameters like cutting velocity and undeformed chip thickness as well as the workpiece and tool material properties. Hence, for difficult to cut materials like temperature resistant super alloys as Inconel 718 tool wear is a crucial factor for tool cost as well as attainable workpiece quality and needs to be monitored during the process [1,2]. There are two major categories for measuring tool wear: direct and indirect [3]. Indirect methods using dynamometers, accelerometers, acoustic emission and current sensors have been widely used for continuous condition monitoring [4–6]. In recent years, Deep Learning (DL) applies to indirect condition monitoring in cutting operations through imaging of time series data and processing this with Convolutional Neural Network (CNN) architectures [7,8]. The functionality of indirect approaches is usually restricted to a certain experimental setup, on a defined machine tool in a defined process. A general solution to the tool condition problem has not been developed yet. Direct measurement of wear on cutting tool edges uses optical sensors which tend to give higher accuracy compared to indirect methods [3]. Nevertheless, uncertainties arise from the measurement process and the interpretation of image pixel data by human operators who decide on a relevant wear form and take a measure, like the width of flank wear land VB [9–11]. Feature detectors for image processing,

10.1016/j.promfg.2020.05.134

like sobel, canny and the active contour method [12–14], are widely applied in literature to detect tool wear on cutting tool edges [15–18]. Another approach is the use of machine learning methods solely or in combination with feature detectors [19,16,20,21]. Classic Computer Vision (CV) algorithms are transparent, power efficient and optimized for specific tasks, while Deep Learning (DL) methods can be used for versatile environments, given the training data reflects the variance [22]. Some typical disturbances in an industrial environment with metal cutting processes are changing light exposure, different coating colors, changing orientation, blur, cold welded chips, dirt and changing tool geometries. A simple example demonstrating the trade-off between the two approaches is shown in Section 3.

Deep learning models for image processing tend to generalize well when enough data is present for the training process. Also these models have proven to defeat traditional approaches in the image processing challenge ImageNet since 2012 [23]. Augmentation methods allow to enlarge the database while bringing artificial variations into the dataset, such as orientation, light conditions, contrast, etc. making the models more robust to changes in the acquisition environment. Tool wear detection is a texture recognition task rather than an object recognition task. A recent study reveals, that CNNs trained on the ImageNet dataset, which contains e.g. cat pictures, are biased towards recognizing textures and not object shapes as previously thought [24]. This means CNNs probably have the ability to recognize texture variations coming from wear phenomena on metal cutting tools. The first occurrence of a deep learning approach for tool wear classification with a VGG-16 architecture yields 96 % precision rate in differentiating four wear forms and a mean absolute percentage error in wear measurement of less than 5 % using traditional image processing methods [25,26].

In this paper, a method for tool type classification and a method for semantic segmentation for tool wear area detection in microscopic images is proposed. Two trained networks with different tasks are presented:

1. An image classification CNN for identification of different cutting tool types.
2. Wear area detection using a Fully Convolutional Network (FCN) namely the U-Net architecture [27].

We test if the application of a heterogeneous dataset that consists of different tool types and is not collected under fixed conditions concerning light, camera position and magnification results in trained networks that are capable of inferring their capabilities on known and unknown tool images. No traditional CV feature detectors occur prior to the DL models to make the approach invulnerable to changing image acquisition conditions that might occur in an industrial environment next to or even within machine tools.

The following sub-sections contain information on the technologies used in the proposed approaches to give a starting point for readers not familiar with the topic of DL. Persons with good knowledge about DL for image processing tasks may jump to section 2.

Nomenclature

API	Application Programming Interface
ADAM	Adaptive Moment Estimation
BN	Batch Normalization
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
FC	Fully-Connected Network
FCN	Fully Convolutional Network
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IoU	Intersect over Union
ML	Machine Learning
NN	Neural Network
OP	Overall Pixel accuracy
R-CNN	Regions with CNN features
ReLU	Rectified Linear Unit

1.1. Neural Networks

The perceptron, also called artificial neuron, is the simplest Neural Network (NN) existent. It emerged as an artificial abstraction of a real neuron and takes one or more input numbers, denoted x, connected via weights, denoted w, to the neuron itself which contains a summation and an activation function, generating an output, y [28]. The bias term, b, shown in Figure 1 may shift the activation function left or right on the x-axis to allow offsetting it to positive or negative values. The perceptron in the figure has an input layer and a hidden layer that functions as output layer at the same time. Equation 1 gives the mathematical expression of a perceptron.

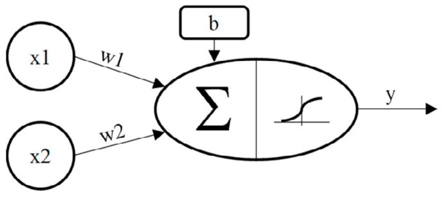


Figure 1: Schematic representation of a perceptron (or artificial neuron)

$$y = f(\sum_i w_i x_i + b) \quad (1)$$

The multilayer perceptron is a wiring of several perceptrons, into a network with at least one hidden and one output layer, each having possibly several perceptrons in parallel. A single perceptron can only learn simple tasks, but for problems that are more complex multilayer perceptrons are required. A NN with two or more hidden layer is called a Deep Neural Network (DNN), see Figure 2.

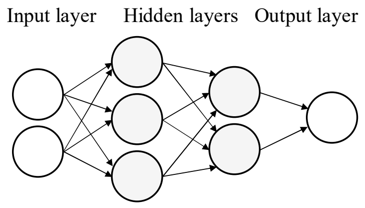


Figure 2: Schematic representation of a Fully-Connected DNN

For simplification, each circle shown below represents an artificial neuron with summation and activation function to process the weighted inputs shown as arrows.

During the network training, with the backpropagation algorithm, one row of data is passed as input through the network [29]. The produced output is compared to the true output yielding an error. The error is propagated back through the network, layer by layer, updating the weights to the amount they contribute to the error. This way all rows of the dataset train the network repeatedly to produce an abstract representation of the data through weights in the NN.

For image processing tasks the above-mentioned, so-called Fully-Connected networks (FC) can be applied but they have a major downside concerning training time. A low-resolution input greyscale image of 32x32 pixels with 1024 neurons in the first layer and the same amount in two hidden layers leads to more than two million trainable parameters considering weights and biases. Scaling this up to relevant architectures with several layers and higher image resolution, the approach gets infeasible. Therefore, FC networks are not scalable for image processing tasks.

1.2. Convolutional Neural Networks

CNNs have applications in object detection, text recognition, pose estimation and many more [30]. They contain several different hidden layer types that bring a solution for the efficiency problem described above. The first CNN applied for digit recognition called LeNet-5 takes 32x32 greyscale images, has roughly 340,000 connections but only 60,000 trainable parameters [31]. An explanation how this is possible is briefly described in the following paragraphs.

The **Convolution Layer** extracts features from an image, e.g. lines and dots, and compresses the image. Using a filter, also called kernel, which slides along the input image, a smaller representation of an image is created.

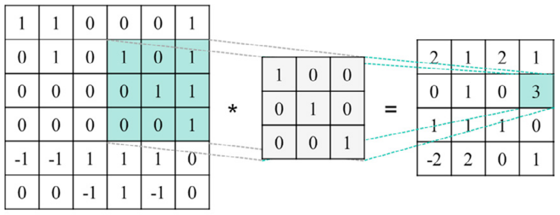


Figure 3: Schematic representation of convolution operation

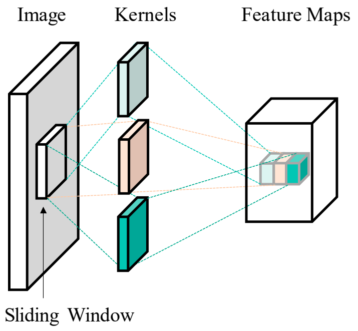


Figure 4: Schematic representation of kernels and feature maps

Applying a 3x3 filter with 1x1 stride, which means one pixel each step, on a 6x6 image yields an output, called feature map, of size 4x4 that means a complexity reduction of more than 50%. Handcrafting of kernels is possible, exemplarily the diagonal line detection in Figure 3. CNNs learn filters during the training process automatically in context of the network task, respectively the training data. In a CNN there are several filters applied in each convolutional layer leading to several feature maps stacked upon each other, see Figure 4. Due to the kernel sizes, closely located pixels relationship is preserved by the convolution operation, which is beneficial for image data where context of nearby pixels holds information about the content.

Pooling reduces the dimension of data through combination of several neuron outputs into one neuron in the subsequent layer. It usually applies after the Convolutional Layer. Figure 5 shows a max pooling operation with stride 2x2 that means a filter movement into one direction always skips a pixel. Another existent type is average pooling where, as the name suggests, an average is taken of the respective numbers within the filter window.

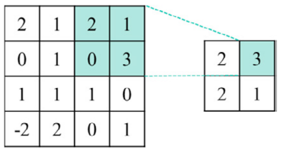


Figure 5: Schematic representation of a pooling operation

With regularization methods, models get more robust and learn more effectively. One approach to this is **Dropout**, see Figure 6, where neurons outputs are deleted randomly. Another widely used and more modern regularization method

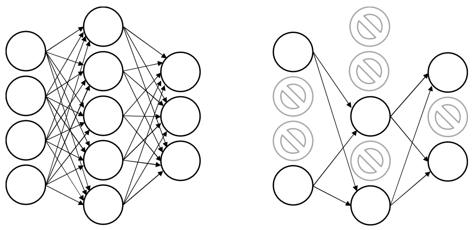


Figure 6: Schematic representation of dropout operation

is Batch Normalization (BN) that adjusts and scales neuron outputs to mean of zero and standard deviation of one [32].

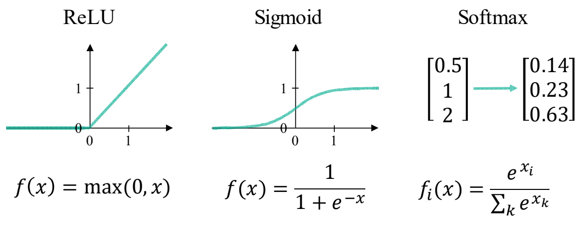


Figure 7: Activation funtions, graphical representation and formula

Activation function layers are applied after hidden or outputs layers of a network. They calculate weighted sums of inputs and biases. In CNNs the most commonly used activation function following a hidden layers is the **Rectified Linear Unit** (ReLU) due to its performance improvements compared to nonlinear activations like **Sigmoid**, its fast computation and its property of preventing vanishing gradients problem which otherwise increases training time [33,34].

Other layers in CNNs are **Flattening Layer** to reshape 3D arrays into 1D vectors, **Fully Connected Layer**, compare 1.1, to learn non-linear combinations of the features coming from the convolutional layers and **Softmax Layer** to normalize into a probability distribution, see Figure 7 and Figure 8.

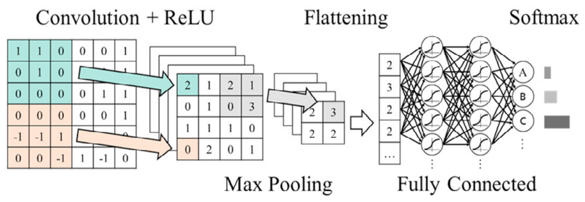


Figure 8: Schematic representation of a simple CNN architecture

1.3. Encoder-Decoder Networks for Semantic Segmentation

With the popularity of DL in recent years, many semantic segmentation problems were addressed using architectures, like CNNs. This approach outperforms others in terms of accuracy and efficiency. Semantic segmentation means that, instead of classifying an image or an object in an image, each pixel in the image is labelled. This enables scene understanding for autonomous driving and analysis of biomedical images for identification of pathological structures [35,36].

The general architecture for segmentation tasks is an encoder, which is often a pre-trained CNN for image classification and a decoder network that classifies each pixel from the features learnt by the encoder. An example for this approach to semantic segmentation is Regions with CNN feature (R-CNN) that performs the task based on object detection results [37]. Another approach to semantic segmentation is Fully Convolutional Networks (FCN) [38]. FCNs take, other than classical CNNs, arbitrary sized images since they do not make use of Fully-Connected layers. FCNs have two parts: a downsampling path to capture contextual information and an upsampling path to recover spatial information.

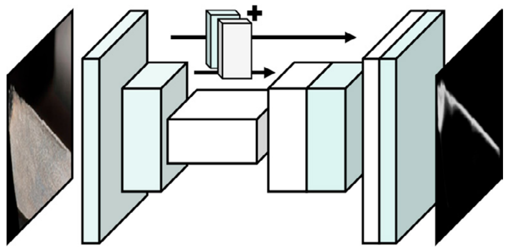


Figure 9: Schematic representation of Encoder-Decoder structure with skip connections for semantic segmentation

The boxes in the conceptual Figure 9 of a segmentation network represent feature maps produced by layers such as convolution plus activation, pooling, upsampling or deconvolutional layers. The arrows represent skip connections that concatenates two equal sizes feature maps from the downsampling and upsampling path. This is necessary to merge contextual and spatial information from various resolutions [38].

2. Materials and Methods

2.1. Hardware and Software

For NN training a Lenovo workstation was used, see specifications in Table 1.

Table 1: PC Hardware specifications for NN training

Name	Specification
Device name	ThinkStation P920
GPUs	2x NVIDIA Quadro P4000
CPUs	2x Xeon Gold 5118

The Graphics Processing Units (GPU) accelerate NN training compared to Central Processing Units (CPU) mainly due to their higher memory bandwidth. That is, GPUs can process large amounts of data at the same time. NVIDIA, a GPU manufacturer, provides an application-programming interface (API) named CUDA to enable using GPUs efficiently for general tasks like DL with support for several DL libraries as in Table 2.

Table 2: Software for image labelling, augmentation, NN training, maths and plotting

Version	Task
Labelme 3.16.1	Label classes on images using polygons
Python 3.7.1	Programming language

Keras 2.3.1	NN library
Tensorflow 1.15.1	ML and DL library
Imgaug 0.3.0	Image augmentation library
Numpy 1.17.3	Large matrix support library
OpenCV 4.1.2	Computer vision library
Matplotlib 3.1.2	Plotting library

Apart from the Python programming language and DL libraries, an open source software called labelme applies for creating the image masks with information about the occurrence of wear on the tool. The created masks are called ground truth because they reflect the true answer about the wear localization. More information on the use of image augmentation library imgaug may be found in section 2.3.

2.2. Database

For ML tasks it is common practice to split the database into several parts. For the training process of the NN the main part of the database applies, the training data. A test dataset, which is split off of the training data, remains for testing the network on unseen images. This means test data comes from the same data pool as training data, but is not shown to the network during training process. The inference dataset consist of images of tools that are not part of training or test data.

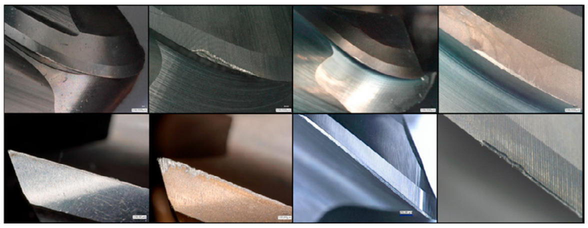


Figure 10: Sample of training data. The first row contains ball end mill images, the second row contains end mill images.

The image data used in this work consists ball-end mill, end mill, insert and drill tools as training data. All images were taken using a Keyence VHX-6000 series microscope (Keyence Corporation, Japan). Table 3 contains information on tool geometry, image count and magnification levels of the training datasets. A sample of the training images for NN training can be found in Figure 10.
To prepare the data for training of a FCN, a pixel-level label process is necessary. Figure 11 shows the processing sequence from original image of a ball end mill cutting edge to ground truth mask that is used for training.

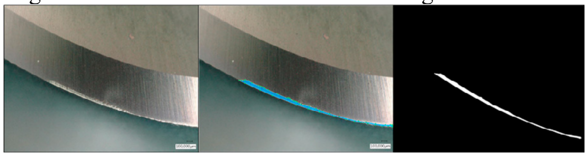


Figure 11: Original image (left), Image during label process (middle), generated ground truth mask (right)

Table 3: Specifications of training and test database with image count

Tool geometries	Training data	Magnification levels and strength
End mill	50	1 (x100)
End mill	50	2 (x20, x150)
End mill corner radius	50	2 (x100, x200)
Ball end mill	50	3 (x30, x50, x100)
Ball end mill	50	3 (x50, x100, x200)
Insert	50	2 (x70, x120)
Insert	50	2 (x100, x150)
Drill	50	1 (x100)

In the inference dataset, see Figure 12, there are images of tools that were not included in the training or test dataset.

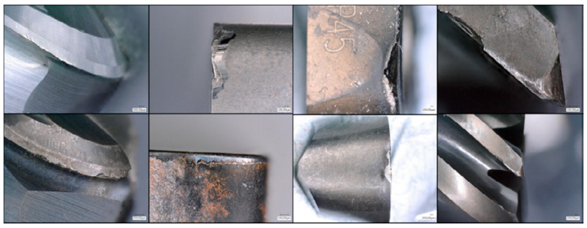


Figure 12: Sample of inference data containing ball end mill, end mill and inserts.

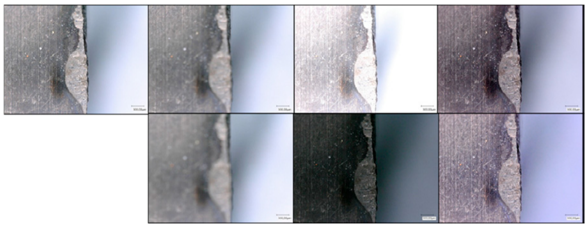


Figure 13: Sample of inference data. The first column shows the original image, second column with moderate and strong blur, third column with high and low light exposure, AR and HDR in the fourth column.

As visible in Figure 13, different capturing settings were applied to bring more variance to the inference images. Specifically, each tool was exposed to following settings using the Keyence microscope:

- High-dynamic-range (HDR) mode and anti-reflection (AR) mode (contrast changes and removed reflections)
- Strong and moderate blur (out of focus point)
- Higher and lower light exposure

This method yields a simulation of disturbances expectable in an industrial environment to evaluate the robustness of developed models towards changing image-recording conditions due to light variations and inaccuracies or inconsistencies of an image acquisition system. Figure 13 shows the effect of different Keyence image acquisition settings on a specimen from the inference dataset.

2.3. Data Augmentation

The performance of DL models relies on the amount and quality of the underlying data. More data usually leads to improving results as well as avoidance of overfitting, i.e. modelling data too close leading to poor generalization capabilities. In domains where data is sparse, data augmentation applies to replace the original dataset with an altered version of itself. Augmentation techniques for image datasets include mirroring, rotating, stretching, squeezing, compressing, colorizing, blurring and many more [39].

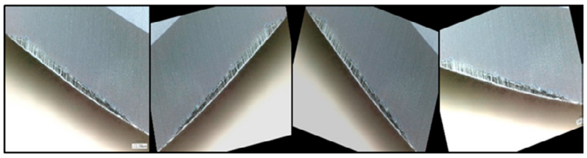


Figure 14: Original image (left) and three augmented images

The images of the training dataset were augmented using the techniques listed in Table 4. An example for augmented versions of an image are shown in Figure 14. After augmentation, the training dataset has 3000 images.

Table 4: Augmentation methods applied to data using imgaug library

Augmentation technique	Settings
Flip	0.5 probability
Multiply	0.9 to 1.1
Rotate	-90 to 90 degrees
Blur (gaussian)	mu:0, sigma:1
Contrast Normalisation	0.9 to 1.1

2.4. Accuracy Metric for Semantic Segmentation

Current literature applies the accuracy metric Intersect over Union (IoU), also known as Jaccard index [40], to assess and compare segmentation models and their capabilities [38,27,41,42]. IoU, see Figure 15, measures the common pixels between ground truth, i.e. manually labelled data, and the predicted mask divided by the union of both.

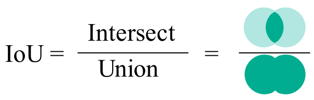


Figure 15: Schematic representation of evaluation metric Intersect over Union

The metric is superior to reporting the correctly classified Overall Pixels (OP) accuracy since the correctly classified background of images biases this metric towards very high values when working with images that have a rather small class representation [43]. Figure 16 shows this bias exemplarily with a tool wear image and its wear prediction.

A predicted mask with deviations visible to the human eye yields 99 % OP accuracy. Even with the mask being completely removed, the OP stays at high levels. IoU in contrast yields a worthier evaluation of the situation.

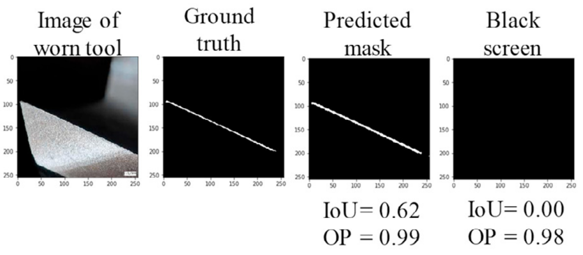


Figure 16: Superiority of IoU over OP as accuracy metric in segmentation

2.5. CNN for Tool Type Classification

A simple CNN architecture design was trained on end mill and ball end mill data for tool type classification, see Figure 17.

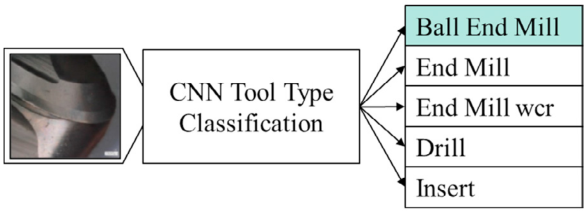


Figure 17: Schematic representation of CNN classifier functionality

Table 5 contains the architecture of this network. Padding is set to same, which means xy-size of feature map stays constant after convolution through adding zeros to the input. A batchsize of ten was used and the network was trained for 30 epochs. The loss function, which measures the mismatch between desired and predicted output during training, is categorical crossentropy:

$$\mathrm{loss} = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \qquad (2)$$

Since this is a multi-class classification, we calculate a separate loss for each class label per observation and sum the result. Here, M is number of classes (drill, end mill, …), log is the natural log, y is a binary indicator (0 or 1) if class label c is the correct classification for observation o and p is the predicted probability observation o belongs to class c. During training, the backpropagation algorithm determines the gradient of the loss function. The algorithm to optimize weights accordingly to minimize the loss is ADAM (Adaptive Moment Estimation), an advanced stochastic gradient descent method. The metric to evaluate network performance is classification accuracy.

Table 5: Structure of CNN for tool type classification

Layer	Output Size	Filter	Strides	Param.
Conv+ReLU	512x512x16	3x3x16	(1, 1)	448

MaxPooling	256x256x16	-	(2, 2)	0
Conv+ReLU	256x256x32	3x3x32	(1, 1)	4640
MaxPooling	128x128x32	-	(2, 2)	0
Conv+ReLU	64x64x32	3x3x32	(2, 2)	9248
MaxPooling	32x32x32	-	(2, 2)	0
Conv+ReLU	16x16x64	3x3x64	(2, 2)	18496
MaxPooling	8x8x64	-	(2, 2)	0
Conv+ReLU	4x4x128	3x3x128	(2, 2)	73856
MaxPooling	2x2x128	-	(2, 2)	0
Dropout	2x2x128	0.3	-	0
Flatten	512	-	-	0
FC with ReLU	128	-	-	65664
FC with ReLU	64	-	-	8256
FC with ReLU	32	-	-	2080
FC with ReLU	16	-	-	528
FC with softmax	5	-	-	85

2.6. *U-Net for Semantic Segmentation*

U-Net is a FCN architecture based on CNN that learns to segment images in an end-to-end setting which means it takes in a raw image and puts out a segmentation map or mask [27], see Figure 18.

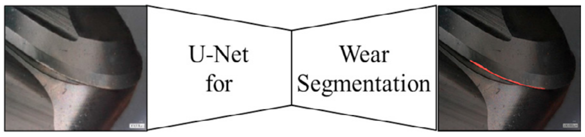


Figure 18: Schematic of U-Net semantic segmentation functionality

The U-Net architecture consists of a large number of different operations, compare section 1.2 and 1.3, illustrated by the small arrows in Figure 19. The input image is fed into the network at the beginning, then the data is propagated through the network along all possible paths and at the end the segmentation map comes out. Each blue box corresponds to a multi-channel feature map. The xy-size and the number of featured channels are denoted in the figure. Most of the operations are convolutions followed by a nonlinear activation function. In detail, there is a standard 3x3 convolution followed by a ReLU. The next operation in the U-Net is a max pooling operation which reduces the xy-size of the feature map as illustrated by a red downward arrow. The max pooling operation acts on each channel separately and propagates the maximum activation from each 2x2 window to the next feature map. After each max pooling operation the number of featured channels are increased by a factor of 2. All these sequence of convolutions and max pooling operations result in a spatial contraction where there is a gradual increase of "what is in the image" and at the same time decrease of "where is it in the image". CNNs for classification tasks end after the contraction and map all features to a single output vector. U-Net has an additional expansion path to create a high-resolution segmentation map. This expansion path consists of a sequence of up-convolutions and concatenation with the corresponding high resolution features from the contracting path. This up-convolution uses a learned kernel to map each feature vector to the 2x2 pixel output window again followed by a nonlinear activation function that outputs the segmentation map. Parameters used in the U-Net training process for semantic segmentation are given in Table 6.

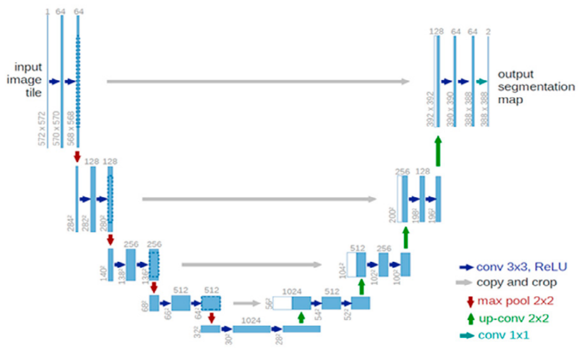


Figure 19: Original U-Net architecture [38]

Table 6: U-Net training parameters

Parameter	Value
Image size	512x512x3
Image format	.jpg
Batch size	2
Epochs	200
GPUs	1
Trainable Parameters	1,941,105
Loss	Binary Crossentropy
Optimizer	ADAM
Metric	IoU coefficient
Training / validation / test	0.8 / 0.1 / 0.1
Pretrained weights	no

Two different approaches were chosen for detection of worn area on cutting tool edges: One-for-all, which means one network that is trained on all data to perform segmentation on arbitrary cutting tool types. One-for-each, which means that for each tool type one specialized network applies. The two approaches see Figure 20, lead to six networks that were trained with similar settings, as in Table 6, but with different database:

- Complete database with images (One-for-all)
- Ball end mill dataset (One-for-each)
- End mill dataset (One-for-each)
- End mill with corner radius dataset (One-for-each)
- Insert dataset (One-for-each)

- Drill dataset (One-for-each)

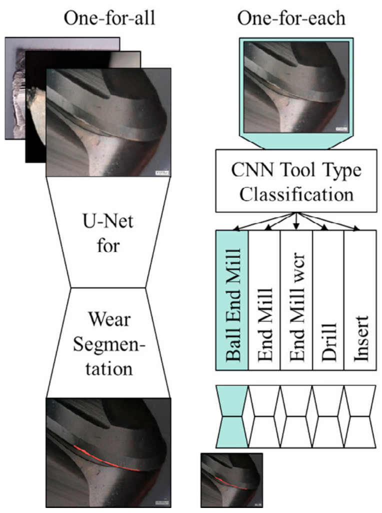


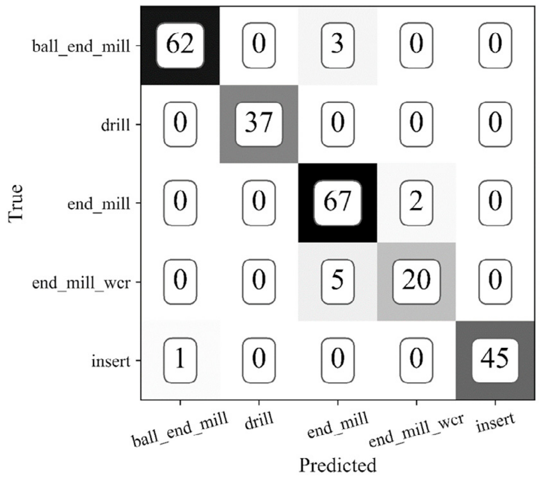
Figure 20: Visualization of One-for-all and One-for-each approach with segmentation networks illustrated as funnels

3. Results

Following chapter splits into the network for tool type classification and the semantic segmentation networks for wear area detection.

3.1. CNN for Tool Type Classification

The simple CNN model described in section 2.5 fulfils the intended task. Datasets of ball end mill, end mill, end mill with corner radius, insert and drill tools are used as the tool-types. Test data counting 242 images yield an accuracy of 95.6 %. The respective confusion matrix is displayed in Figure 21.



True \ Predicted	ball_end_mill	drill	end_mill	end_mill_wcr	insert
ball_end_mill	62	0	3	0	0
drill	0	37	0	0	0
end_mill	0	0	67	2	0
end_mill_wcr	0	0	5	20	0
insert	1	0	0	0	45

Figure 21: Confusion matrix for test dataset of tool type classification network with accuracy of 95.6 %

3.2. U-Net for Semantic Segmentation of Tool Wear

The scatter plots and sample images shown in this section refer to the One-for-all approach. Figure 22 shows the distribution of IoU values for tool wear segmentation. The figure contains the IoU scatter plot of the training and validation data, as well as test and inference data. Validation data indicates overfitting when validation data accuracy

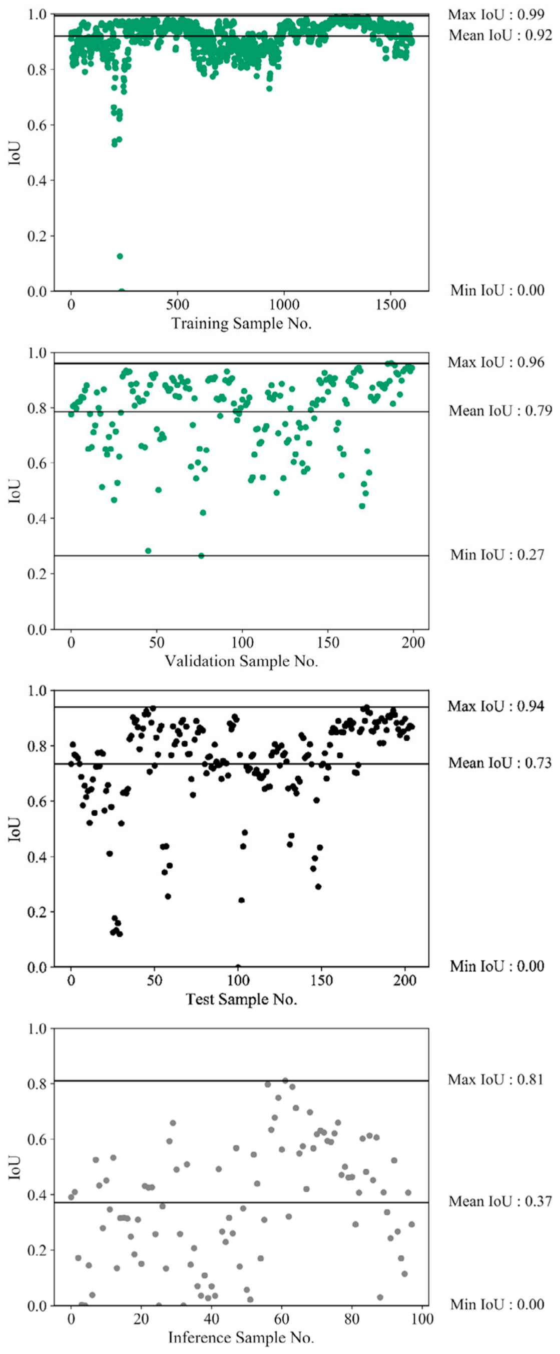


Figure 22: Distribution of IoU over all images in the respective datasets for One-for-all approach

starts to decrease and diverge from training accuracy. Weak overfitting can be observed, but test data accuracy of roughly the same level as validation data accuracy still proves the ability of the DL approach to generalize. When the network is exposed to completely unknown data with perturbations as shown in Figure 13, i.e. the inference data, the accuracy halves compared to test data accuracy.

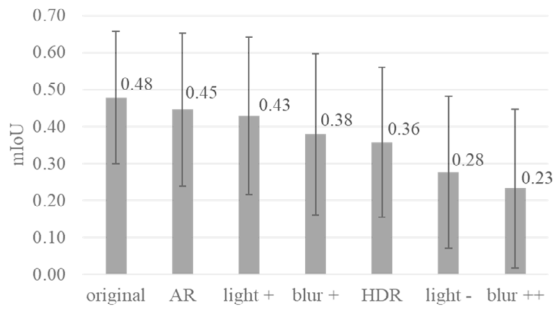


Figure 23: Mean and standard deviation of IoU metric for different capturing settings of inference data (One-for-all network)

Figure 23 contains the mean IoU of inference data with different capturing settings. Low light and highly blurred images pose the greatest challenge for the network. Figure 24 shows a sample image from the inference dataset with their respective mask and IoU score.

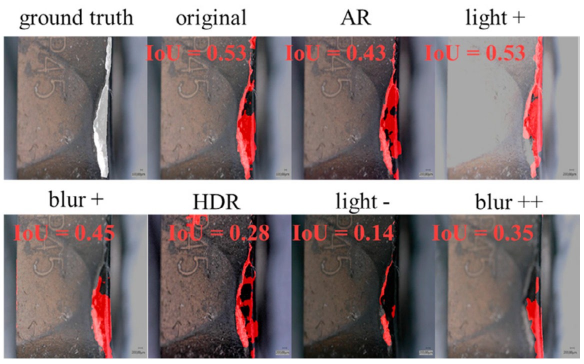


Figure 24: Different capturing settings with predicted mask and IoU coefficient of an insert from the inference dataset (One-for-all network)

Table 7 contains the IoU evaluation metrics of all six approaches. For each network the minimum, maximum and mean IoU are given for the training, validation and test data.

Table 7: Results of U-Net for semantic segmentation given as IoU scores

Network	IoU training min/**mean**/max	IoU validation min/**mean**/max	IoU test min/**mean**/max
all	0.00/**0.92**/0.99	0.27/**0.79**/0.96	0.00/**0.73**/0.94
ball end mill	0.79/**0.91**/0.99	0.00/**0.66**/0.92	0.41/**0.70**/0.86
end mill	0.00/**0.90**/0.99	0.17/**0.66**/0.90	0.08/**0.71**/0.93
end mill wcr	0.85/**0.95**/0.99	0.63/**0.83**/0.93	0.76/**0.87**/0.94
insert	0.86/**0.95**/0.99	0.37/**0.71**/0.89	0.31/**0.72**/0.93
drill	0.85/**0.94**/0.98	0.63/**0.83**/0.93	0.76/**0.87**/0.94

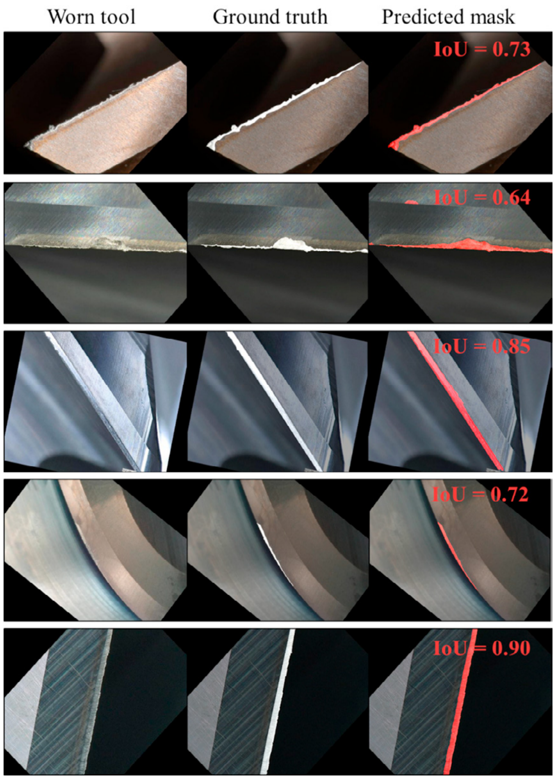


Figure 25: Test data samples. Original image (left), human made ground truth mask (middle) and prediction of network (right) with IoU

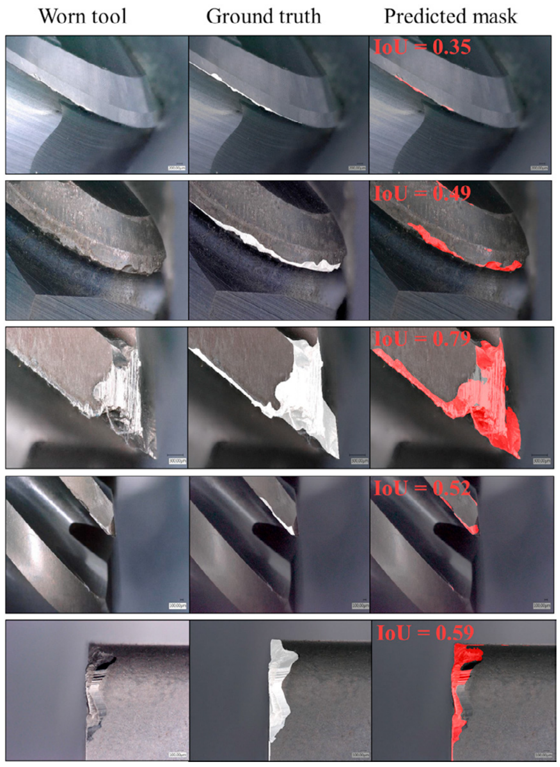


Figure 26: Inference data samples. Original image (left), human made ground truth mask (middle) and prediction of network (right) with IoU

Figure 25 displays sample images of the augmented test dataset for the One-for-all network. Each figure contains the original tool image on the left, the label mask in the middle and the prediction on the right. In accordance with the IoU of 0.73 of the test data, the network detects the worn area reliably. For inference data, i.e. different images than in the training, validation or test dataset, the results are worse, as visible in Figure 26. This is in accordance with the mean IoU of 0.37 showing again that IoU is a good metric to evaluate quality of semantic segmentation results. The predictions in the figure are erroneous at large worn areas, as well as at edges or surfaces with textural damage that resembles wear.

As discussed previously, the DL approach is more robust compared to traditional computer vision algorithms against disturbances, such as changing light exposure. In Figure 27 an image of an insert from [44] is taken and the proposed traditional CV approach is applied. Except for minor differences, the results of the traditional approach and DL seem very similar.

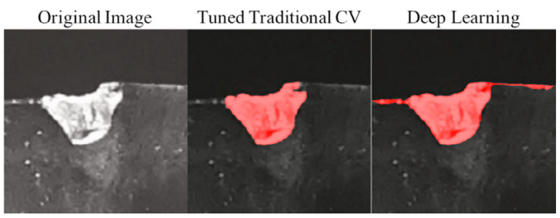


Figure 27: Wear segmentation of insert from literature with traditional CV and proposed DL approach

However, Figure 28 shows the same tuned CV approach, as well as the DL approach, applied to another similar looking image of our dataset with two variations concerning light exposure. The DL approach shows better generalization capabilities as well as robustness towards changing light conditions.

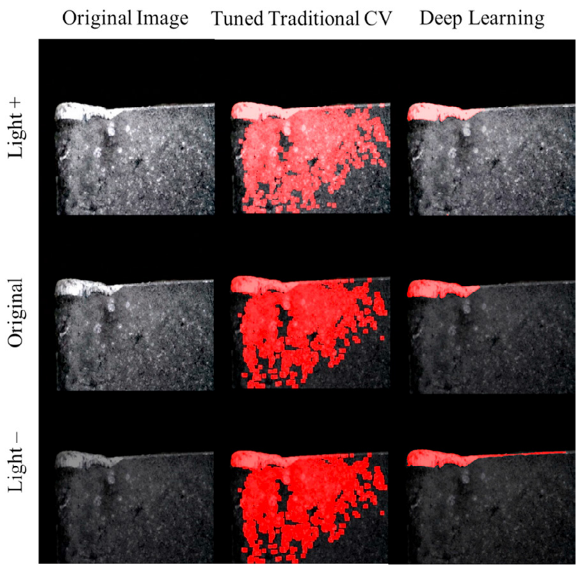


Figure 28: Wear segmentation of insert from own data with increased and decreased light exposure. Comparison of tuned traditional CV approach and DL approach

4. Discussion

In this paper an approach to tool type classification and an approach to tool wear detection for cutting tools is proposed.

- For tool type classification, ball end mill, end mill, end mill with corner radius, inserts and drills are used as classes to be distinguished by a CNN with a test accuracy of 95.6 %.
- The tool wear detection approach using a FCN, namely U-Net, for semantic segmentation achieves promising results with mean IoU coefficient of 0.73 on test data. Training data consists of 3000 augmented images stemming from 400 raw images. Eight different cutting tool datasets with 50 images each and various levels of magnification make up the heterogeneous raw image database.
- Unknown and disturbed tool data, as in the inference dataset, yields a mean IoU of 0.37 with tendency of the network to misrecognize irrelevant edges and scratches as well as missing out bits of large worn areas. This means a generalization that allows inferring the One-for-all model to make reliable predictions on unknown image data with disturbances could not be reached. Reasonable progress for this goal might be attainable with more training examples.
- Inference capabilities of the segmentation network decrease across tool types for different capturing conditions. Anti-reflection and increased light yields the best performance within the inference dataset with still poor mean IoU coefficients below 0.5, whereas low light and severe blur yields mean IoU coefficients below 0.3, which can be considered failed predicition. A standard deviation of mean IoU across capturing methods is roughly 0.2 which means a higher number of images is required in inference dataset to increase the certainty of this finding.
- Data preparation requires more time and attention for segmentation networks compared to the classification network since the ground truth masks must be made manually with great care. Coarse masking might be applicable with large datasets. In that case the test data must still be labellend as accurate as possible to allow a meaningful evaluation of network performance with the IoU coefficient.
- Approaches to train models only on one tool type, i.e. One-for-each approach, yield similar results to the One-for-all approach although only a fraction of data applies for training. For further work it might make sense to train a classifier model as proposed here to distinguish tool types and then pass over the segmentation task to a network specialized for this very tool type.

Apart from an automated method to evaluate tool wear within or outside the machine tool using microscopes to make practical decisions on, for example tool changes, the method may be applied in machine tools that generate monitoring models. The tool wear detection method will allow an acceleration and objectification in modelling tool life, in creating tool condition monitoring models or other process monitoring models as well as in cutting tool design. Specifically labelling high frequency sensor data with wear values to create indirect condition monitoring models using this automated detection approach is a promising idea for future work which may enable automated generation of monitoring models for scientific and industrial applications.

The method might transfers well to other cutting or manufacturing processes where tool degradation takes place and impairs the product quality.

The proposed tool wear detection method can be extended with further NNs to classify wear types [45,26] and localize wear phenomena with bounding boxes [46] to segment relevant cutouts of the image with higher pixel quality.

Another possibility is reading out the scale automatically using CNNs and taking measurements of wear criteria like width of flank wear land VB [47] in a combinatorial approach with traditional CV techniques.

Further work could include applying NNs for deblurring and super-resolution to tackle image quality issues with machine tool integrated low cost microscopes [48,49].

Methods for model improvement using unlabelled data or artificial data generation using Generative Adversarial Networks (GAN) could help accelerating the data generation and increasing the database artificially [50,51].

This paper contributes to the vision of a fully automated cutting tool wear analysis method using machine tool integrated microscopes in the scientific and industrial environment as depicted schematically in Figure 29.

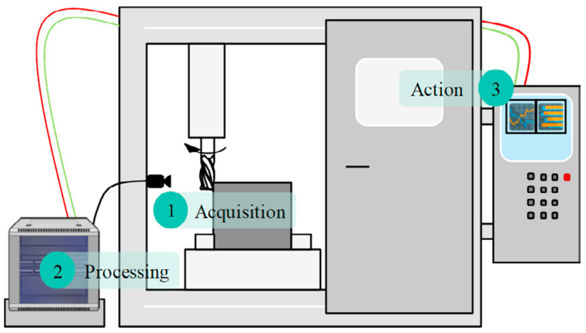


Figure 29: Schematic representation of a fully automated tool wear analysis system within machine tools

Acknowledgements

The Federal Ministry of Transport and Digital Infrastructure (BMVI) has founded this publication through the 5G-IndustryCampus Europe project.

References

[1] Ezugwu, E.O., Wang, Z.M., Machado, A.R., 1999. The machinability of nickel-based alloys: a review. Journal of Materials Processing Technology 86 (1-3), 1–16.
[2] Wang, B., Liu, Z., 2018. Influences of tool structure, tool material and tool wear on machined surface integrity during turning and milling of titanium and nickel alloys: a review. Int J Adv Manuf Technol 98 (5-8), 1925–1975.
[3] Jeon, J.U., Kim, S.W., 1988. Optical flank wear monitoring of cutting tools by image processing. Wear 127 (2), 207–217.
[4] Abellan-Nebot, J.V., Romero Subirón, F., 2010. A review of machining monitoring systems based on artificial intelligence process models. Int J Adv Manuf Technol 47 (1-4), 237–257.
[5] Drazen Veselovac, 2013. Process and Product Monitoring in the Drilling of Critical Aero Engine Components. Dissertation, Aachen.
[6] Zhou, Y., Xue, W., 2018. Review of tool condition monitoring methods in milling processes. Int J Adv Manuf Technol 96 (5-8), 2509–2523.
[7] Gouarir, A., Martínez-Arellano, G., Terrazas, G., Benardos, P., Ratchev, S., 2018. In-process Tool Wear Prediction System Based on Machine Learning Techniques and Force Analysis. Procedia CIRP 77, 501–504.
[8] Martínez-Arellano, G., Terrazas, G., Ratchev, S., 2019. Tool wear classification using time series imaging and deep learning. Int J Adv Manuf Technol 104 (9-12), 3647–3662.
[9] International Institution for Production Engineering Research, 2004. Wörterbuch der Fertigungstechnik: Trennende Verfahren. Springer Berlin Heidelberg, 298 pp.
[10] International Standard. ISO 3685: Tool Life Testing with single point Turning. International Organization for Standardization, 53 pp.
[11] International Standard. ISO 8868-2: Tool Life Testing in Milling - Part 2: End Milling. International Organization for Standardization, 32 pp.
[12]Canny, J., 1986. A Computational Approach to Edge Detection. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (6), 679–698.
[13]Kanopoulos, N., Vasanthavada, N., Baker, R.L., 1988. Design of an image edge detection filter using the Sobel operator. IEEE J. Solid-State Circuits 23 (2), 358–367.
[14]Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. Int J Comput Vision 1 (4), 321–331.
[15]Alegre, E., Alaiz-Rodríguez, R., Barreiro, J., Ruiz, J., 2009. Use of contour signatures and classification methods to optimize the tool life in metal machining. Estonian J. Eng. 15 (1), 3.
[16]D'Addona, D.M., Ullah, A.M.M.S., Matarazzo, D., 2017. Tool-wear prediction and pattern-recognition using artificial neural network and DNA-based

computing. J Intell Manuf 28 (6), 1285–1301.
[17]Kurada, S., Bradley, C., 1997. A review of machine vision sensors for tool condition monitoring. Computers in Industry 34 (1), 55–72.
[18]Moldovan, O., Dzitac, S., Moga, I., Vesselenyi, T., Dzitac, I., 2017. Tool-Wear Analysis Using Image Processing of the Tool Flank. Symmetry 9 (12), 296.
[19]D'Addona, D.M., Teti, R., 2013. Image Data Processing via Neural Networks for Tool Wear Prediction. Procedia CIRP 12, 252–257.
[20]Dhanachandra, N., Manglem, K., Chanu, Y.J., 2015. Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm. Procedia Computer Science 54, 764–771.
[21]Xiong, S.C., Dong, L.P., Wen, D.H., 2010. Tool Wear Image Segmentation Based on Markov Random Field Model. AMR 102-104, 600–604.
[22]Mahony, N.O.', Campbell, S., Carvalho, A., Harapanahalli, S., Velasco-Hernandez, G., Krpalkova, L., Riordan, D., Walsh, J., 2020. Deep Learning vs. Traditional Computer Vision. 2194-5357 943.
[23]Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60 (6), 84–90.
[24]Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W., 2019. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.
[25]Simonyan, K., Zisserman, A., 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition, 14 pp. http://arxiv.org/pdf/1409.1556v6.
[26]Wu, X., Liu, Y., Zhou, X., Mou, A., 2019. Automatic Identification of Tool Wear Based on Convolutional Neural Network in Face Milling Process. Sensors (Basel, Switzerland) 19 (18).
[27]Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation, 8 pp. http://arxiv.org/pdf/1505.04597v1.
[28]ROSENBLATT, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 65 (6), 386–408.
[29]LeCun, Y., 1988. A Theoretical Framework for Back-Propagation. Proceedings of the 1988 Connectionist Model Summer School, 21–28.
[30]Aloysius, N., Geetha, M., 2017. A review on deep convolutional neural networks: 6th-8th April 2017, Melmaruvathur, India, 588–592.
[31]Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.
[32]Ioffe, S., Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 11 pp. http://arxiv.org/pdf/1502.03167v3.
[33]Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S., 2018. Activation Functions: Comparison of trends in Practice and Research for Deep Learning, 20 pp. http://arxiv.org/pdf/1811.03378v1.
[34]Vinod Nair, Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines.
[35]Lundervold, A.S., Lundervold, A., 2019. An overview of deep learning in medical imaging focusing on MRI. Zeitschrift fur medizinische Physik 29 (2), 102–127.
[36]Mennatullah Siam, Mostafa Gamal, Moemen Abdel-Razek, Senthil Yogamani, Martin Jagersand, Hong Zhang. A Comparative Study of Real-Time Semantic Segmentation for Autonomous Driving.
[37]He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN, 12 pp. http://arxiv.org/pdf/1703.06870v3.
[38]Long, J., Shelhamer, E., Darrell, T. Fully convolutional networks for semantic segmentation, 3431–3440.
[39]Shorten, C., Khoshgoftaar, T.M., 2019. A survey on Image Data Augmentation for Deep Learning. J Big Data 6 (1), 1106.
[40]Jaccard, P., 1912. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. New Phytol 11 (2), 37–50.
[41]Zhang, N., Li, J., Li, Y., Du, Y. Global Attention Pyramid Network for Semantic Segmentation, 8728–8732.
[42]Zhong, Z., Lin, Z.Q., Bidart, R., Hu, X., Daya, I.B., Li, J., Wong, A., 2019. Squeeze-and-Attention Networks for Semantic Segmentation, 10 pp. http://arxiv.org/pdf/1909.03402v2.
[43]Csurka, G., Larlus, D., Perronnin, F. What is a good evaluation measure for semantic segmentation?, 32.1-32.11.
[44]Thakre, A.A., Lad, A.V., Mala, K., 2019. Measurements of Tool Wear Parameters Using Machine Vision System. Modelling and Simulation in Engineering 2019 (4), 1–9.
[45]Liu, W., Wei, L., Sharpnack, J., Owens, J.D., 2019. Unsupervised Object Segmentation with Explicit Localization Module, 10 pp. http://arxiv.org/pdf/1911.09228v1.
[46]Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement, 6 pp. http://arxiv.org/pdf/1804.02767v1.
[47]Bonechi, S., Andreini, P., Bianchini, M., Scarselli, F., 2019. Weak Supervision for Generating Pixel-Level Annotations in Scene Text Segmentation, 10 pp. http://arxiv.org/pdf/1911.09026v1.
[48]Lu, Z., Chen, Y., 2019. Single Image Super Resolution based on a Modified U-net with Mixed Gradient Loss, 18 pp. http://arxiv.org/pdf/1911.09428v1.
[49]Zhang, S., Zhen, A., Stevenson, R.L., 2019. Deep Motion Blur Removal Using Noisy/Blurry Image Pairs, 10 pp. http://arxiv.org/pdf/1911.08541v2.
[50]Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A., 2016. Image-to-Image Translation with Conditional Adversarial Networks, 17 pp. http://arxiv.org/pdf/1611.07004v3.
[51]Sivanesan, U., Braga, L.H., Sonnadara, R.R., Dhindsa, K., 2019. Unsupervised Medical Image Segmentation with Adversarial Networks: From Edge Diagrams to Segmentation Maps, 16 pp. http://arxiv.org/pdf/1911.05140v1.