Online Chatter Detection using Self-Evolving Automated Machine Learning Fusion

by

MohammadHossein Rahimi

B.Sc., Sharif University of Technology, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

March 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Online Chatter Detection using Self-Evolving Automated Machine Learning Fusion

submitted by M. Hossein Rahimi in partial fulfillment of the requirements for

the degree of Master of Applied Science

in Mechanical Engineering

Examining Committee:

Dr. Yusuf Altintas, Mechanical Engineering
Supervisor

Dr. Bhushan Gopaluni, Chemical and Biological Engineering
Supervisory Committee Member

Dr. Xiaoliang Jin, Mechanical Engineering
Supervisory Committee Member

Abstract

Computer Numerical Controlled (CNC) Milling is used to remove excess metal from a blank to produce the final shape of the workpiece. One limitation of high material removal rates with reduced cost in machining operations is self-excited vibrations called chatter. Chatter results in poor surface finish, damage to the workpiece and machine tool's spindle, and causes accelerated tool wear. Chatter detection and prevention have been one of the major fields of study in manufacturing to improve the quality and productivity of machining operations.

This research proposes a self-evolving, online method to detect and avoid chatter in milling operations by fusing deep learning with the knowledge of chatter theory. The process is monitored by collecting vibration data during machining. Windows of data are converted into Short-time Fourier Transforms and processed through a Convolution Neural Network to identify five machining states: Air cutting, entrance to and exit from the cut, stable cut, and unstable cut with chatter. A base state detection model is built by modifying and training AlexNet architecture using experimental data with known states. A specialized Deep Learning architecture is designed based on the base model, using an Automated machine learning process, with high state detection accuracy and low complexity in mind.

In parallel to the machine learning model, chatter detection with a physics-based model is executed to increase the robustness and accuracy of chatter detection. The forced vibrations which always occur in milling are removed by Kalman filter, and the occurrence of chatter is detected using an energy-based method. The hybrid system detects the chatter with a 98.8% success rate. The system has a built-in online self-improving capability. The system stops the machining process and commands a new spindle speed to force the machine to operate in chatter-free zone.

Lay Summary

Computer Numerical Controlled (CNC) machines are widely used in the machining industry. The motion of machine along the commanded tool path and as well as the rotation of the spindle which carries the tool is governed by the CNC, and the final shape of the product is produced through metal removal.

The cost and machining time in aerospace and die & mold are quite significant. When a machine experiences unstable chatter, the resulting vibrations lead to high forces and poor surface finish which may destroy the part, tool, spindle and costly parts.

This thesis presents a method to detect and suppress chatter vibration by infusing the physics of the cutting process and Artificial Intelligence. The proposed method can detect the chatter and identifies the best spindle speed to avoid chatter during machining.

Preface

I was the lead investigator for the work presented in Chapters 3, 4 and 5. I planned the experimental design, conducted experimentation, developed the algorithms, and completed results analysis. Dr. Yusuf Altintas was my supervisor.

A version of chapters 3, 4 and chapter 5 has been presented at a conference. I presented the material at the British Columbia Artificial Intelligence Showcase conference. Additionally, the manuscript "On-line chatter detection in milling using machine learning" was written by M. Hossein Rahimi, Hoai Nam Huynh, and Dr. Yusuf Altintas.

Table of Contents

List of Tables

List of Figures

List of Symbols

h_j	Regenerative chip thickness for j-th tooth
θ_j	Angular immersion angle for j-th tooth
$F_{r,j}$, $F_{t,j}$	Radial and tangential forces for j-th tooth
a_p	Depth of cut
q	Modal displacement vector
K_t, K_r	Tangential and radial cutting force coefficients
$a_{xx}, a_{xy}, a_{yx}, a_{yy}$	Time-varying periodic directional force coefficients
$Q_{c,n}$, $\tilde{Q}_{c,n}$	Complex coefficient vectors and their conjugates at the n-th harmonic
Q	Process noise covariance matrix
$X_{p,n}$, $X_{c,n}$	Amplitude of n-th periodic and chatter vectors
λ	Process noise to measurement noise ratio
F	Force vector
K	Kalman gain vector
H	Measurement matrix
M	Number of band-pass filters
R	Measurement noise covariance
D	Lag parameter
$\hat{q}_k$	Estimated state vector
ψ	Nonlinear energy operator (NEO)
ψ_d	Nonlinear energy operator discrete form
y_c	Chatter component

$\hat{Y}_p$	Energy of the periodic part of the signal
$\hat{Y}_{p,n}$	Amplitude of the n-th harmonic
η	Energy integration factor
τ	Time delay
Φ	Phase
Ψ	Non-linear energy operator
EN_c	Chatter energy
EN_P	Periodic energy
ER	Energy ratio
ω_t	Tooth passing Frequency
ω_s	Spindle Frequency
ω_c	Chatter Frequency
σ	Standard Deviation
$\tilde{\mu}_3$	Skewness
f_s	Sampling frequency
M	Windowing function length
L_w	STFT window size
L_o	STFT overlap size
$\mathrm{ReLU}(x)$	Rectified linear unit
σ_β	Softmax function
$\hat{f}$	Surrogate function
μ_Q	Accusation function

$p(state)$	Probability of each machining state determined by the machine learning algorithm
p_{state}	Probability of each state
P_C	Accumulative Probability of chatter
P_S	Accumulative Probability of stable process
$B^{(l)}$	Output of l-th Convolutional Block
$K^{(l)}$	Weights matrix of Kernel of l-th Convolutional Block
$C^{(l)}$	Output of l-th Convolutional Layer
w^l	Weights matrix of Neural Network l-th layer
$w_{i,j}^l$	Weight of the connection of neuron i from layer $l-1$ and neuron j from layer l
b_l	Biases matrix of layer l
b_j^l	Bias of layer l and neuron j
FL	Focal loss
s_i	Network design hyperparameters set for i-th iteration
W_P	Weights matrix of loss penalty
$a_i^{(l)}$	Activation of neuron i of layer l
C_i	Convolutional block i
n_c	Number of Convolutional blocks
N_i	Neural network hidden layer i
n_N	Number of the neural networks hidden layers
$L(\hat{\theta})$	Likelihood of $\hat{\theta}$

T	Number of training instances
$k_{\text{complexity}}$	Complexity of network
Y_c	Frequency-domain signal of the non-periodic part of the vibration signal
a_{th_i}	Peak threshold of i-th point of discrete-time
f_{sup}	Suppression factor
$th_{\text{Certainty}}$	Certainty threshold
th_{ER}	Self-evolution energy ratio certainty threshold
K_p	Energy-based influence factor
α	Learning rate
ϵ	Offset denominator
m_i, v_i	Parameter gradients and their squared

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AutoML	Automatic Machine Learning
CNN	Convolutional Neural Network
DDE	Delay Differential Equations
DFT	Discrete Fourier Transform
ESA	Energy Separation Algorithm
FFT	Fast Fourier Transform
FL	Focal Loss
FRF	Frequency Response Function
HQC	Hannan–Quinn information criterion
IIR	Infinite impulse response
LSTM	Long Short-term Memory
ML	Machine Learning
NAS	Neural Architecture Search
NEO	Teager-Kaiser Nonlinear Energy Operator
ODE	Ordinary differential equation
SDOF	Single Degree of Freedom
STD	Standard Deviation
STFT	Short-term Fourier transform

Acknowledgments

I would like to express my gratitude to my supervisor, Professor Yusuf Altintas, for the confidence he placed in me by accepting me in the Manufacturing Automation Lab and for his continued support and guidance throughout my master of applied science research.

To my friends and peers in the Manufacturing Automation Laboratory, thank you for your technical and moral support and thank you for all the shared laughs and conversations. I would like to thank Dr. Hoai Nam Huynh for his valuable inputs during the process of writing my thesis.

Finally, I would like to thank my magnificent family. I am incredibly grateful to my parents. Without the inspiration, drive, and support that they have given me, I might not be the person I am today.

Chapter 1: Introduction

Machining is a subtractive manufacturing process where a cutting tool moves relative to a workpiece and removes material to form the desired shape. Machining processes are critical operations in manufacturing parts with final shapes. The productivity rates and dimensional accuracy of the machined parts are of great importance. Violating dimensional tolerances during production scraps the parts, which are costly and disruptive to the manufacturing chain in the factories.

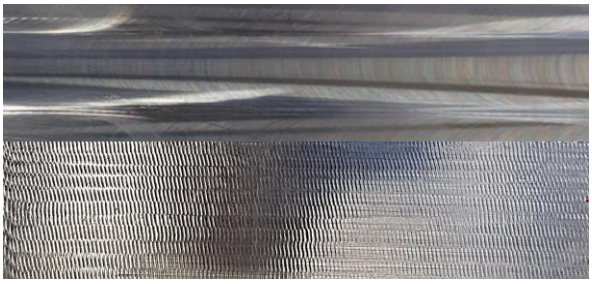

Figure 1 Stable Cut (top) versus Chatter (bottom) Surface Quality

Self-excited vibrations in machining, namely the chatter, is the most important limitation for high material removal and dimensional accuracy of the machined parts [1]. As shown in Figure 1, an unstable machining process with chatter leaves vibration marks on the workpiece and results in poor surface quality. During chatter, the amplitude of unstable vibration may increase exponentially leading to excessive chip thickness hence large cutting forces. Chatter does not only result in poor surface quality and excessive noise but can overload the spindle bearings and cause accelerated tool wear.

Chatter can be avoided by selecting the depth of cuts and cutting speeds within the stable zone during process planning before the part is machined. The stable cutting conditions can be predicted as a function of the Frequency Response Function (FRF) of the machine at the tool-workpiece structures at their contact zone, material properties, and tool geometry. However, there are always

uncertainties in material properties and FRF of the machine structure which may change under operating conditions. Therefore, an online chatter detection and avoidance system during the machining process is necessary to run the machine in stable conditions.

As shown in Figure 2, the milling process commonly exhibits three primary sets of forces. The stable, forced vibrations occur at tooth passing frequency and its harmonics. Chatter, the unstable vibrations, occur close to one of the natural frequencies. There are also sidebands that are spread at the multiples of tooth passing frequency away from the chatter frequency. Having multiple vibration sources with frequencies scattered over various frequency ranges makes it difficult to differentiate between the stable and unstable cut. Moreover, the machining system has numerous states such as entry to and exit from the workpiece, which excite transient vibrations at the natural frequencies of the machine which resemble chatter.

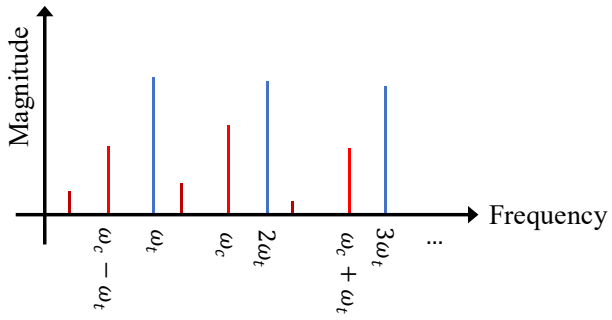

Figure 2 Machining Forces for a Sample Process with ω_c Chatter Frequency Illustration

Data-driven methods could be used for chatter detection to account for time-varying system parameters and compensate for the measurement inaccuracies by processing vibration data collected during machining. However, it is essential to ensure the robustness and versatility of the data-driven method to be used for chatter detection.

This thesis presents an online chatter detection method by infusing a data-driven machine learning method, with a physics-driven method to achieve robustness and accuracy. As shown in Figure 3, the proposed method consists of a machining state detection using convolutional neural networks and a chatter detection algorithm based on the physics of milling running in parallel. Both methods use the vibration signal as their input.

The first algorithm is a self-evolving machine learning method, capable of differentiation between stable and unstable cut and detecting machining states, i.e. tool-workpiece engagement. The process of data acquisition, data handling, plausibility assessment, signal processing, signal features extraction, machine learning model selection, and architecture design are described. A substantial number of cutting experiments are performed that resulted in a database of cutting experiments. The experiment database is directly used for the assessment of machining state detection using machine learning. Furthermore, the vibration signal is used for architecture design and model training.

A deep learning model called Convolutional Neural Network (CNN) is used in the thesis. The network architecture, corresponding to the model's hyper-parameters, is designed using a process known as Automated Machine Learning (AutoML). The architecture design procedure starts with a transfer learning method as the initial input to the designed loop. During the designed AutoML algorithm's execution process, a Bayesian optimization algorithm seeks the best architecture in the defined search space. The resultant architecture is used for online machining state detection. The method uses the time-domain vibration signal as the input and converts it to the frequency domain to perform cutting state detection. It outputs the probability of each machining state.

An automatic procedure is designed to evolve the machine learning model’s performance while it is running online. It uses statistical methods to enhance the system's performance as it functions and observes more machining cases.

The second simultaneously running method is an enhanced version of a previously developed physics-driven method, called online energy-based chatter detection. The method takes vibration signal as the input, as well, and output the chatter to periodic energy ratio, which is used as an indicator of chatter. It detects chatter by removing all the periodic components using a Kalman filter. Then by transforming the signal to the frequency domain and setting a statistical threshold to group the chatter peaks. It finds the eligible peaks based on certain criteria and calculates the chatter energy. It uses the ratio of the chatter energy to the accumulative chatter and stable energy. A higher ratio indicates a higher chance of instability.

The algorithm calculates the probability of chatter from the machine learning method and the physics-based method's energy ratio. The cutting states detected by the machine learning algorithm aids the algorithm to avoid false chatter detection. When the chatter is detected, the algorithm stops the feed, changes the spindle speed and resumes the cutting until chatter is avoided. If the chatter is impossible to avoid due to violation of physical limits, the operator is alerted to stop the operation.

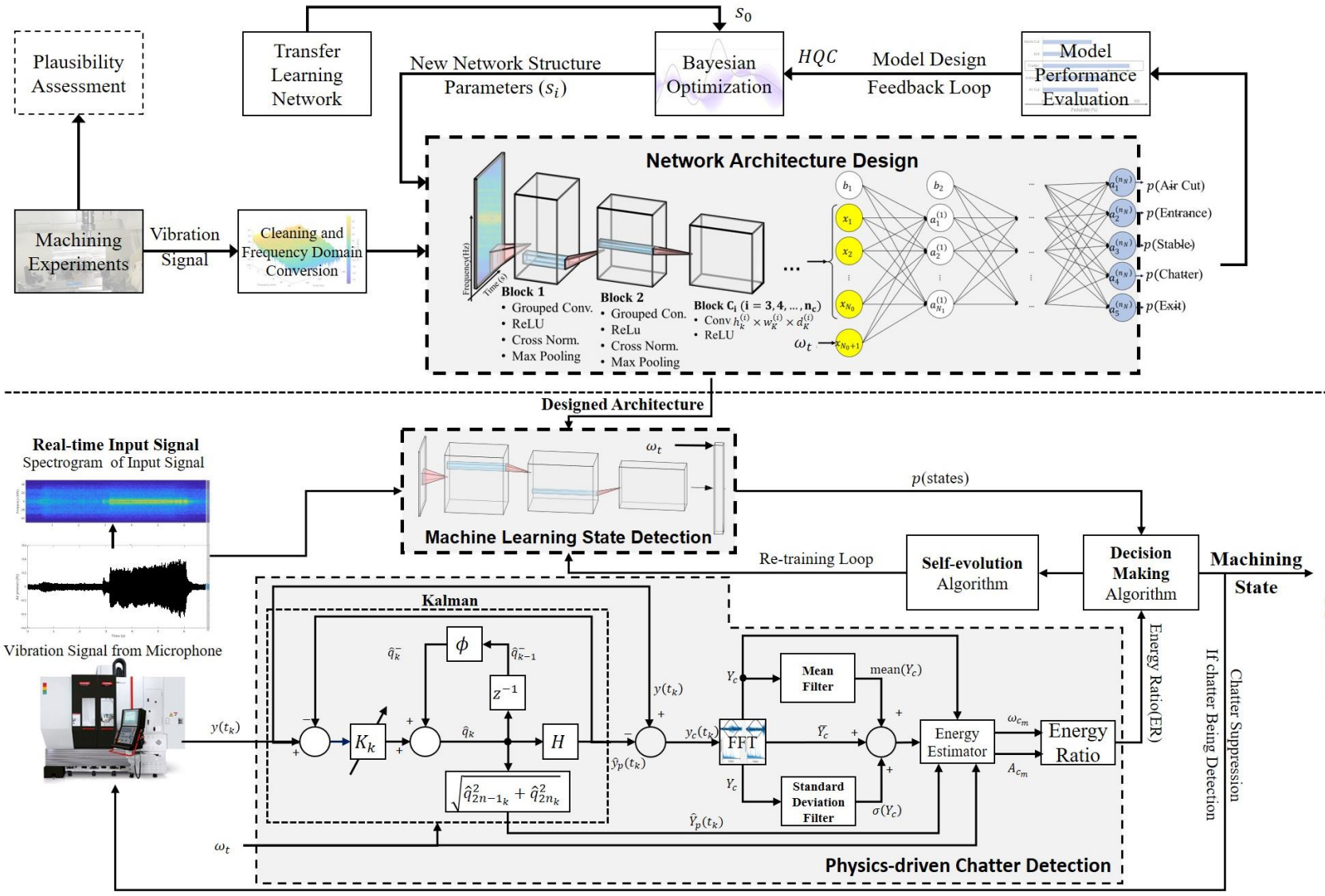

Figure 3 System Overview

The thesis is structured as follows. Chapter 2 discusses the state of the art literature in chatter detection based on both physics and data-driven methods. It includes a brief review of the recent development in artificial intelligence used in the field of machining stability.

Chapter 3 elaborates on the process of designing and training the machine learning model used for online machining state detection. The chapter starts with the experiment design process and data handling. The experimental data used to train a machine learning model is presented. The trained Model is used as the initial state of an automated machine learning architecture design. The automated design process is developed and implemented to increase the chatter detection accuracy, and decrease the model complexity and computing power.

Chapter 4 presents the machine learning model's fusion with the physics-based model. The first section of the chapter goes over the proposed improvements to the physics-based method, which increases its accuracy and robustness. Furthermore, the chapter explains the algorithm to fuse the two methods and describes the approach taken towards making the algorithm self-evolving.

Chapter 5 presents the experimental validation of the models. The chapter evaluates the performance of the methods individually, as well as accumulatively. Chapter 6 concludes the thesis with final remarks and future work.

Chapter 2: Literature Review

There have been numerous attempts to predict and detect chatter before and during the machining process. Prior art involving online chatter detection can be roughly categorized into two following approaches, physics-driven methods, which use metal cutting dynamics to detect chatter vibration; and data-driven methods, which leverage the experimental data to find patterns and detect chatter. This work suggests a machine learning method for machining state detection, a physics-driven method for chatter detection, and a method to link two methods and leverage the knowledge of metal cutting to improve the data-driven method's performance.

In this chapter, prior research related to chatter detection, both using data-driven and physics-based methods, and automated machine learning architecture design are surveyed. The first section reviews the dynamics of metal cutting. Section 2.2 elaborates on the previous attempts on chatter detection and briefly explain the energy-based chatter detection method which is adopted here as a physics-based method. Section 2.3 presents the machine learning methods and their utilization in machining dynamics. Finally, section 2.4 demonstrates the conclusion derived from the literature and presents where this work stands among them.

2.1 Dynamic of Metal Cutting

The dynamic model of milling with relative tool-workpiece vibrations is shown in Figure 4, alongside a block diagram description of the system. In the below figure, x_w and y_w represent the position of the workpiece and x_t and y_t reperesent the position of the tool.

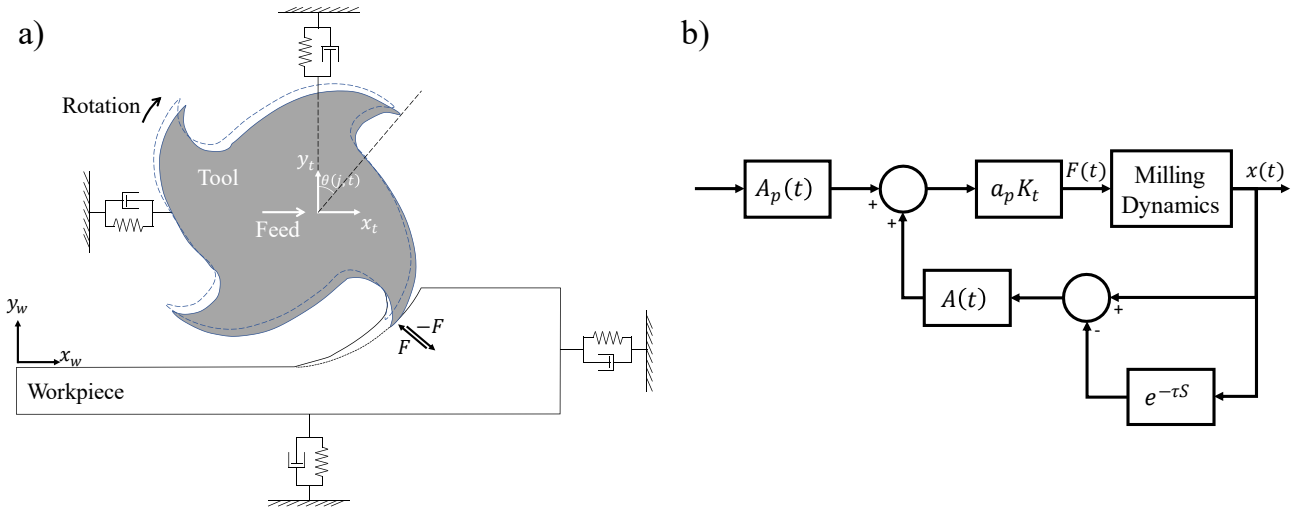

Figure 4 Orthogonal Cutting a) Dynamic Scheme, b) Block Diagram

The cutting forces act at the tool and workpiece contact in the opposite direction. The forces are proportional to the static and regenerative chip thickness. The regenerative chip thickness is given by [2],

$$h_j(t) = c \sin\theta_j(t) + [x(t) - x(t-\tau)] \sin\theta_j(t) + [y(t) - y(t-\tau)] \cos\theta_j(t) \quad (1)$$

Where the subscript $j \in \mathbb{N}$ is the tooth index, and $c[\mathrm{mm/rev/tooth}]$ is the feed rate per tooth. The term $\tau[s]$ represents the time delay and equal to the tooth passing period $(60/(\omega[\mathrm{rev/min}]))$ for a regular pitch tool. Additionally, $\theta_j[\mathrm{rad}]$ is the angular immersion angle of tooth j,

$$\theta_j(t) = 2\pi\left(\frac{\omega}{60}t + \frac{j-1}{N}\right) \quad (2)$$

The radial force $(F_{r,j})$ and tangential force $(F_{t,j})$ of tooth j for cutting material with K_r and K_t radial and tangential cutting coefficients, respectively, is expressed as [3],

$$\begin{cases} F_{r,j}(t) = g_j(t)\rho_j a_p K_r h_j(t) \\ F_{t,j}(t) = g_j(t)\rho_j a_p K_r h_j(t) \end{cases} \tag{3}$$

Where a_p is the depth of cut, ρ_j is the run-out factor [4], and $g_j(t)$ is an indicator of whether the tooth j is in the cut or not as,

$$g_j(t) = \begin{cases} 0 : \text{If } j\text{th tooth is not in cut at time } t \\ 1 : \text{If } j\text{th tooth is in cut at time } t \end{cases} \tag{4}$$

Similarly, in the x-y frame, the forces are given by,

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \frac{1}{2}\boldsymbol{a_p K_t}.\underbrace{\begin{bmatrix} a_{xx}(t) & a_{xy}(t) \\ a_{yx}(t) & a_{yy}(t) \end{bmatrix}}_{2A(t)} \begin{bmatrix} \boldsymbol{x(t)} - \boldsymbol{x(t-\tau)} \\ \boldsymbol{y(t)} - \boldsymbol{y(t-\tau)} \end{bmatrix} + \overbrace{\boldsymbol{a_p K_t c}\underbrace{\begin{bmatrix} a_{xx}(t) \\ a_{yx}(t) \end{bmatrix}}_{A_p(t)}}^{\text{Static forces}(F_s)} \tag{5}$$

Where $a_{xx}(t), a_{xy}(t), a_{yx}(t), a_{yy}(t)$ are time-varying, periodic directional force coefficients expressed as [3],

$$\begin{aligned} a_{xx}(t) &= -\sum_{j=1}^{N_t} g_j(t)\rho_j \left[\sin 2\theta_j(t) + \frac{K_r}{K_t}\left(1 - \cos 2\theta_j(t)\right)\right] \\ a_{xy}(t) &= -\sum_{j=1}^{N_t} g_j(t)\rho_j \left[1 + \cos 2\theta_j(t) + \frac{K_r}{K_t}\sin 2\theta_j(t)\right] \\ a_{yx}(t) &= \sum_{j=1}^{N_t} g_j(t)\rho_j \left[1 - \cos 2\theta_j(t) - \frac{K_r}{K_t}\sin 2\theta_j(t)\right] \\ a_{yy}(t) &= \sum_{j=1}^{N_t} g_j(t)\rho_j \left[\sin 2\theta_j(t) - \frac{K_r}{K_t}(1 + \cos 2\theta_j(t))\right] \end{aligned} \tag{6}$$

The modal displacement vector of the system, with a size equal to the total number of vibration modes of tool and workpiece, could be represented as the summation of its periodic component $q_p(t)$ and perturbation $q_c(t)$ as below,

$$q(t) = q_p(t) + q_c(t) \tag{7}$$

For a machine with a proportionally damped dynamics cutting under a stable condition, the equation of motion takes the following ordinary differential equation (ODE) form,

$$\begin{aligned} &\ddot{\boldsymbol{q}}(\boldsymbol{t}) + [\boldsymbol{2\xi\omega_n}]\dot{\boldsymbol{q}}(\boldsymbol{t}) + [\boldsymbol{\omega_n^2}]\boldsymbol{q}(\boldsymbol{t}) \\ &\qquad = \boldsymbol{U^T a_p K_t A(t) U[q(t) - q(t-\tau)]} + \boldsymbol{U^T F_s(t)} \end{aligned} \tag{8}$$

Where U is the mode shape matrix, ω_n is natural frequency, and F_s is the static forces from equation (5). For a stable system with no perturbation, the steady-state vibrations as follows,

$$\begin{aligned} &\ddot{\boldsymbol{q}}_p(\boldsymbol{t}) + [\boldsymbol{2\xi\omega_n}]\dot{\boldsymbol{q}}(\boldsymbol{t}) + [\boldsymbol{\omega_n^2}]\boldsymbol{q_p}(\boldsymbol{t}) = \boldsymbol{U^T F_s(t)} \xrightarrow{\textbf{ODE Solution}} \boldsymbol{q_p(t)} \\ &\qquad = \sum_{\boldsymbol{n=1}}^{\infty} \boldsymbol{Q_{p,n} e^{i(n\omega_s t)}} \end{aligned} \tag{9}$$

Given equations (8) and (9), the self-excited vibrations are found from the solution of the Delayed Differential Equation (DDE) [5].

$$\boldsymbol{q_c(t) = q(t) - q_p(t) = \sum_{n=-\infty}^{\infty} \left(Q_{c,n} e^{i(\omega_c + n.\omega_t)t} + \widetilde{Q}_{c,n} e^{-i(\omega_c + n.\omega_t)t} \right)} \tag{10}$$

Where $\omega_c[\mathrm{rad/s}]$ is the chatter frequency, and $Q_{c,n}$ and $\tilde{Q}_{c,n}$ are respectively the complex coefficient vectors and their conjugates at the n-th harmonic.

Relative tool-workpiece displacement vector $x(t)$ is determined by the steady-state solution of equation (8) as shown below,

$$x(t) = \sum_{n=1}^{\infty} X_{p,n}\cos\left(n\omega_s t + \Phi_{p,n}\right) + \sum_{n=-\infty}^{\infty} X_{c,n}\cos\left((\pm\omega_c + n\omega_t)t + \Phi_{c,n}\right) \quad (11)$$

Where $X_{p,n}$, and $X_{c,n}$ are respectively the amplitude of n-th periodic and chatter vectors. Moreover, $\Phi_{p,n}$, and $\Phi_{c,n}$ are the phase of n-th periodic and chatter vectors, respectively.

2.2 Chatter Prediction and Detection using Physics-driven methods

Chatter stability of milling has been studied extensively in the past few decades [6]. Tlusty and Polacek presented the stability laws requiring the frequency response function of both the machine tool and the workpiece, cutting coefficients of the work material, and tool geometry [7]. Their work resulted in stability lobes, which set a theoretical limit on the maximum stable depth of cut for each given spindle speed. However, the uncertainties in the frequency response function measurements and its corresponding variables, the system's position, and time-varying dynamics may still lead to chatter, even in a theoretical stable zone based on stability lobes.

The difficulties in chatter prediction and the machine tool's time-varying parameters resulted in research in online chatter detection and suppression methods. There have been several attempts to detect chatter during the machining process. T. Delio et al. [8] used a microphone signal during the milling process and converted it to the frequency domain to detect chatter. They set a threshold on the power spectrum to detect chatter occurrence. In another art, T. Choi and Y.C.Shin [9] detected chatter online using the fractional patterns they observed in the acceleration signals. They utilized a wavelet-based maximum likelihood estimation algorithm. In attempting to detect chatter,

M. Lamraoui et al. [10] investigated the cyclo-stationary components generated by the machining process.

2.2.1 Online Energy-based Chatter Detection

Energy-based chatter detection [11] is an online chatter detection method that uses milling physics to detect instability in the cut. Despite its high potentials and robustness, false chatter detection happens quite often in energy-based chatter detection, especially during the transient states of cut, such as tool entrance into the workpiece or when it leaves the workpiece, since the machine is excited sue to a condition similar to impact hit and display different vibration contents and sudden changes in the vibration signal.

Energy-based chatter detection identifies chatter by comparing energy levels in the force vibration components and the signal's unstable components. In the energy-based chatter detection method, the periodic part of the signal and its energy are identified using a Kalman filter. The periodic part is subtracted from the whole signal, and the chatter energy is calculated using a bandpass filter bank and a Teager-Kaiser Nonlinear Energy Operator (NEO). Finally, the method compares the two calculated energies to conclude about the stability of the machining process.

The outline of energy-based chatter detection is presented in Figure 5 with a sample input to illustrate the process.

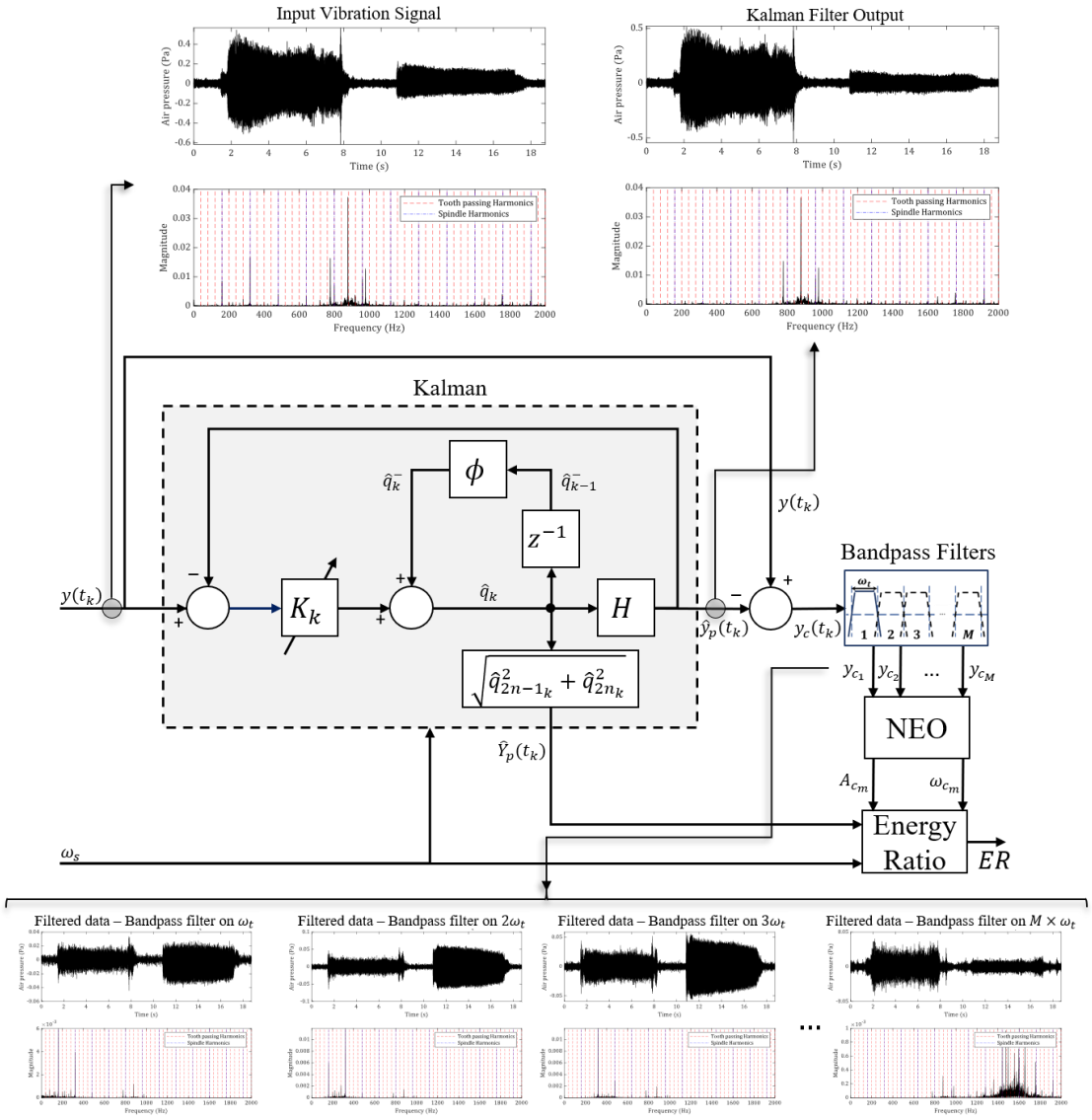

Figure 5 Online energy-based chatter detection overview with a Sample Input to Illustrate the Process

2.2.1.1 Identification of the Periodic Component of Machining Vibrations

Forces during chatter can be decoupled into periodic components due to cutting and forces close to the structural mode frequency excited by regenerative chatter, which are given by equations (9) and (10), respectively.

The measured vibration signal is decoupled into the periodic part that causes forced vibrations and the unstable, chatter components ($y = y_p + y_c$). The assumption is that the process is unstable when the energy of the self-excited vibration term (y_c) dominates the cutting process. The following equation estimates the periodic part of the cutting signal,

$$\boldsymbol{y_p(kT) = \sum_{n=1}^{N} Y_{p,n}(kT)\cos(n\omega_s kT + \phi_{p,n})}, \tag{12}$$

where $Y_{p,n}(kT)$ is the amplitude of the spindle harmonic with frequency $n\omega_s$ ($n = 1, 2, \dots, N$). The state variable representation of equation (12) with N number of harmonics is shown below,

$$\boldsymbol{y(t_k) = \underbrace{[1 \quad 0 \quad \dots \quad 1 \quad 0]}_{H_{1\times 2N}} q_{k_{2N\times 1}} + v_k} \tag{13}$$

Where $q_{k+1_{2N\times 1}}$ represents the state variable representation at discrete time instance of $k+1$, has been expressed as follows,

$$\boldsymbol{q_{k+1_{2N\times 1}} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{2N} \end{bmatrix}_{k+1} = \underbrace{\begin{bmatrix} \phi_1 & 0 & \dots & 0 \\ 0 & \phi_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & & \phi_N \end{bmatrix}}_{\phi_{2N\times 2N}} \underbrace{\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{2N} \end{bmatrix}_k}_{q_{k_{2N\times 1}}} + \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{2N} \end{bmatrix}_k}_{w_{k_{2N\times 1}}}} \tag{14}$$

And ϕ_i is the state transition matrix used to model i-th harmonic ($i = 1, 2, 3, \dots, N$). Let the first and second state be q_{1_k} and $q_{2_k} = A_k \sin(\omega t_k + \phi)$ respectively, at the time $t_{k+1} = t_k + T$ will be derived as,

$$
\begin{aligned}
q_{1_{k+1}} &= A_{k+1}\cos(\omega(t_k+T)+\phi) \\
&= A_{k+1}[\cos(\omega t_k+\phi)\cos(\omega T) \\
&\quad -\sin(\omega t_k+\phi)\sin(\omega T)] \\
q_{2_{k+1}} &= A_{k+1}\sin(\omega(t_k+T)+\phi) \\
&= A_{k+1}[\cos(\omega t_k+\phi)\sin(\omega T) \\
&\quad -\sin(\omega t_k+\phi)\sin(\omega T)]
\end{aligned}
\tag{15}
$$

Hence the state variable is equal to,

$$
\begin{bmatrix} q_{1_{K+1}} \\ q_{2_{K+1}} \end{bmatrix} = \begin{bmatrix} \cos(\omega T) & -\sin(\omega T) \\ \sin(\omega T) & \cos(\omega T) \end{bmatrix} \begin{bmatrix} q_{1_k} \\ q_{2_k} \end{bmatrix} + \begin{bmatrix} w_{1_k} \\ w_{2_k} \end{bmatrix} \Longrightarrow
$$

$$
\phi_n = \begin{bmatrix} \cos(n\omega_s T) & -\sin(n\omega_s T) \\ \sin(n\omega_s T) & \cos(n\omega_s T) \end{bmatrix} \tag{16}
$$

The state vector of the periodic signal (14) has been estimated using below Kalman filter [12],

$$
\begin{cases}
\hat{q}_k^- = \phi \hat{q}_{k-1} \\
P_k^- = \phi P_{k-1}\phi^T + \overset{Q}{\overbrace{\lambda R}} \\
K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \\
\hat{q}_k = \hat{q}_k^- + K_k(s_k - H\hat{q}_k^-) \\
P_k = (I - K_k H) P_k^-
\end{cases} \tag{17}
$$

Where process noise to measurement noise ratio initially is $\lambda = 10^{-6}$.

The chatter part of the signal has been estimated based on the estimated state vector ($\hat{q}_k$) with,

$$
y_c(t_k) = y(t_k) - \hat{y}_p(t_k) = y(t_k) - H\hat{q}_k \tag{18}
$$

Given $\cos^2(\omega t_k) + \sin^2(\omega t_k) = 1$ the amplitude of the n-th harmonic is,

$$
\hat{Y}_{p,n}(t_k) = \sqrt{\hat{q}_{2n-1_k}^2 + \hat{q}_{2n_k}^2} \tag{19}
$$

2.2.1.2 Chatter Component Assessment

Chatter component reconstruction consists of two major steps. First, the signal is filtered to isolate the frequencies in between each two consecutive tooth passing harmonics using a set of bandpass filters. The second step is to use Nonlinear Energy Operator (NEO) [13] and Energy Separation Algorithm (ESA) [14] in order to estimate a measure of the energy of the single component signal in the discrete-time domain.

The signal $\boldsymbol{y_c(t_k)}$ goes through a series of bandpass filters. In order to isolate the chatter part of the signal corresponding to each chatter frequency laying in between tooth passing harmonics, a bank of bandpass filters has been used. The number of filters is equal to the number of tooth passing harmonics(N). The mentioned method results in N output signals, each isolated frequencies in between one of the tooth passing. The mentioned filters have been designed using the Butterworth method with half-power frequencies and infinite impulse response (IIR) of order four. Each filter ith filter have half-power frequencies equal to $(i-1)\omega_t$ and $i\omega_t$.

Nonlinear Energy Operator (NEO) has been introduced by Kaiser [13]; it has been denoted by ψ in this thesis and defined as,

$$\boldsymbol{\psi[x(t)] = \dot{x}(t)^2 - x(t)\ddot{x}(t)} \tag{20}$$

For the below undamped SDOF system,

$$\boldsymbol{m\ddot{x} + kx = 0 \Rightarrow x(t) = A\cos(\sqrt{k/m}\, t + \phi)} \tag{21}$$

where A and ϕ are the amplitude and phase angle of the harmonic motion frequency and $\omega = \sqrt{k/m}$ is the natural frequency of the oscillation. Given $kx = -m\ddot{x}$ the total kinetic and potential energy of the system is given by,

$$\boldsymbol{E(t) = \frac{1}{2}m[\dot{x}(t)]^2 + \frac{1}{2}k[x(t)]^2 = \frac{m}{2}A^2\omega^2} \tag{22}$$

P.Maragos and T.F.Quatieri [14], suggested the following continuous Teager-Kaiser Nonlinear Energy Operators (NEOs) as a measure to track the energy per half-unit mass of the source of oscillation,

$$\begin{cases} \boldsymbol{\psi[x(t)] = \dot{x}(t)^2 - x(t)\ddot{x}(t) = A^2\omega^2} \\ \boldsymbol{\psi[\dot{x}(t)] = \ddot{x}(t)^2 - \dot{x}(t)\dddot{x}(t) = A^2\omega^4} \end{cases} \Longrightarrow$$
$$\boldsymbol{\omega = \sqrt{\frac{\psi[\dot{x}(t)]}{\psi[x(t)]}}, |A| = \frac{\psi[x(t)]}{\sqrt{\psi[\dot{x}(t)]}}} \tag{23}$$

Hence, in the time instance of t_k the corresponding frequency $\omega(t_k)$ and amplitude $\left|A_{t_k}\right|$ could be calculated as below,

$$\begin{cases} \boldsymbol{\theta(t_k) = \omega(t_k)T = arccos\left(1 - \frac{\psi_d[\Delta x_k] + \psi_d[\Delta x_{k+1}]}{4\psi_d[x_k]}\right)} \\ \boldsymbol{\left|A_{t_k}\right| = \sqrt{\frac{\psi_d[x_k]}{1 - \left(1 - \frac{\psi_d[\Delta x_k] + \psi_d[\Delta x_{k+1}]}{4\psi_d[x_k]}\right)^2}}} \end{cases} \tag{24}$$

Given the equation (24), for the chatter frequency and amplitude of the m-th harmonic is given by,

$$\begin{cases} \boldsymbol{\omega_{C_m} = arccos\left(1 - \frac{\psi_d[\Delta y(t_{k-2D})] + \psi_d[\Delta y(t_{k-D})]}{4\psi_d[y_c(t_{k-2D})]}\right)} \\ \boldsymbol{\left|A_{C_m}\right| = \sqrt{\frac{\psi_d[y_c(t_{k-2D})]}{1 - \left(1 - \frac{\psi_d[\Delta y(t_{k-2D})] + \psi_d[\Delta y(t_{k-D})]}{4\psi_d[y_c(t_{k-2D})]}\right)^2}}} \end{cases} \tag{25}$$

Where ψ_d is the generalized discrete form of NEO as below,

$$\begin{cases} \psi_d[\Delta y(t_{k-D})] = \dfrac{\Delta y(t_{k-D})^2 - \Delta y(t_{k-2D})\Delta y(t_k)}{T^2} \\ \psi_d[\Delta y(t_{k-2D})] = \dfrac{\Delta y(t_{k-2D})^2 - \Delta y(t_{k-3D})\Delta y(t_{k-D})}{T^2} \end{cases} \tag{26}$$

The difference operators for each band are as below,

$$\begin{cases} \Delta y(t_{k-D}) = y_{c_m}(t_{k-D}) - y_{c_m}(t_{k-2D}) \\ \Delta y(t_{k-2D}) = y_{c_m}(t_{k-2D}) - y_{c_m}(t_{k-3D}) \\ \Delta y(t_{k-3D}) = y_{c_m}(t_{k-3D}) - y_{c_m}(t_{k-4D}) \end{cases} \tag{27}$$

Where D is the lag parameter introduced by W. Lin [15] and equal to one-quarter of the corresponding m-th chatter frequency,

$$D = \mathbf{round}\left(\frac{1}{4T}\frac{2\pi}{(m-0.5)\omega_t}\right) \tag{28}$$

The resultant frequencies and amplitudes of the harmonics go through a mean filter with a delay equal to spindle frequency.

The energy ratio of the signal is given by,

$$\mathbf{ER} = \frac{\mathbf{EN_c}}{\mathbf{EN_c} + \mathbf{EN_p}} \tag{29}$$

Where EN_p and EN_c are respectively the energy of periodic and chatter part of the signal and could be determined by,

$$\begin{cases} \mathbf{EN_p} = \sum_{n=1}^{N} \left(\widehat{Y}_{p_n}\right)^2 (n\omega_s)^{2\eta} \\ \mathbf{EN_c} = \sum_{m=1}^{M} \left(\widehat{Y}_{c_m}\right)^2 \left(\omega_{c_m}\right)^{2\eta} \end{cases} \tag{30}$$

Where η is the integration factor and determined based on the signal type and $\widehat{Y}_{p_n}$ is given by equation (19).

Based on the energy ratio calculated by equation (29), the stability of the cut will be determined. The energy ratio is commensurate to the chatter energy from equation (30); hence, in this thesis, it has been used as an indicator of the chance of an ongoing unstable cut.

In this thesis, the energy ratio has been used as a measure of the probability of the chatter happening. The details of energy-based chatter detection method used here are presented by Caliskan, Kilic and Altintas in [11].

2.3 Machine Learning and its utilization in Industry and Machining Sciences

Artificial Intelligence (AI) is a field that studies the synthesis and analysis of computational agents that act intelligently [16]. Machine Learning (ML) is a subfield of AI concerned with developing automated methods to learn the patterns in a given set of data with the purpose of being used on future data or other outcomes of interest [17].

In the literature, researchers are trying to use a variety of machine learning methods to predict or detect chatter. M. Postel et al. [18] used deep learning to refine the chatter stability lobes. M. Lamraoui et al. [19] investigated different statistical features of the envelope signal, such as variance, skewness, kurtosis, peak value, root mean square (RMS), clearance factor, crest factor, shape factor, and impulse factor. They found variance as a practical feature for chatter detection. In the mentioned work, a fully connected neural network is trained to detect chatter. The collected data for chatter detection is done by performing slot milling tests on an aluminum workpiece. Zh. Yao et al. [9] proposed an online chatter detection method using support vector machines (SVM) to determine the stability of cut based on the vibration signal's wavelet transform features.

Automated Machine Learning (AutoML) methods reduce the amount of resources and time required to create a well-performing model [20]. C. White et al. [21] showed that the Bayesian optimization can outperform other common AutoML methods, such as random search, regularized evolution, and reinforcement learning, for Neural Architecture Search (NAS). X. He et al. [20] reviewed the recent advancements in the AutoML and provided a literature survey.

Machine learning models are capable of being improved by getting retrained on a new set of data. Ch. Rosenberg et al. [22] suggested that self-training makes it possible for the machine learning systems to improve over time while they are performing. P. Mitra et al. [23] investigate the effectiveness of Bayesian AutoML methods for Physics Emulators.

Ch. Rosenberg [22] explored and evaluated semi-supervised learning and self-training approaches for training a machine learning model on weakly labelled data by conducting experiments.

2.4 Summary

In this chapter, past works of literature regarding chatter detection machine learning architecture design are reviewed. There have been advancements in chatter prediction, before the operation and during the process planning, in the past few decades. Online chatter detection is essential due to inaccuracies in the measurements and time-varying nature of the countless number of machine components.

The past literature in the field is either focused on physics-based chatter detection or data-driven method methods. To compensate for each approach's shortcomings, machine learning combined

with the physics of metal cutting is preferred. However, in most machine learning chatter detection literature, the physics of the process is ignored.

Among the many online chatter detection methods in the literature, an energy-based system is utilized to account for the process's physics most closely [11]. In literature, different machine learning models are proposed for chatter detection. This work has taken an automated approach towards machine learning network design by proposing an automated network design scheme.

Chapter 3: Online Cutting State Detection Using Machine Learning

This chapter proposes a machine learning pipeline for machining state detection as well as two machine learning models, one transfer learning and the other one custom designed architecture, for detecting instability in the cutting process as well as the machining state based upon its vibration input. A machine learning pipeline is a set of consecutive mathematical functions and operations that has a set of inputs, a machine learning network, and outputs of interest. Other than inputs, machine learning model and outputs, a machine learning pipeline commonly consists of a transformation function, which in this case is a signal cleaning and short-time Fourier transform functions. The designed pipeline uses a deep learning model, known as Convolutional Neural Network, for machining state and chatter detection. Deep learning is a subspace of machine learning in artificial intelligence.

In this chapter, the process of designing the deep learning method used for cutting state detection is explained. The outcome of this chapter is a trained deep learning architecture designed specifically for machining state detection.

The chapter is organized as follows. The first section provides an overview of the machine learning pipeline, architecture design, and its training process. Section 2.2 is dedicated to the experiment design process and describing the process of data collection. The collected data are used for deep learning architecture design, training, and evaluation, as well as for parameter tuning. The technical details behind handling the high-volume data collected from sensors during the machining process, and its utilization in the designed machine learning pipeline are discussed. In section 2.3, the details on converting the signal to the frequency domain are discussed. Section 2.4 starts with outlining the deep learning architecture, which is used for machining state detection

using transfer learning. Afterwards, each of the architecture's elements with the intuition behind their use for chatter detection is described. Section 2.5 proposes an approach towards automated CNN architecture design and elaborates on the implementation and utilization of the proposed method for online machining state detection. In the proposed method, a Bayesian optimization algorithm is used as an automated design agent, which is briefly explained in the section. The outcome of section 2.4 is used as the initial input to the optimization loop in this section.

3.1 Machine Learning Training Pipeline

An overview of the step taken towards designing machine learning architecture and training the model is given in Figure 6. It is a detailed depiction of the “Automated CNN Architecture Design” part of Figure 3. Training is a process during which the network parameters get tweaked in a way that minimizes the loss, and the final resultant network parameter could serve its goal, which is machining state detection, in the best way possible. This process's output is a trained machining state detection network capable of determining each cutting state's probability based on the vibration signal input.

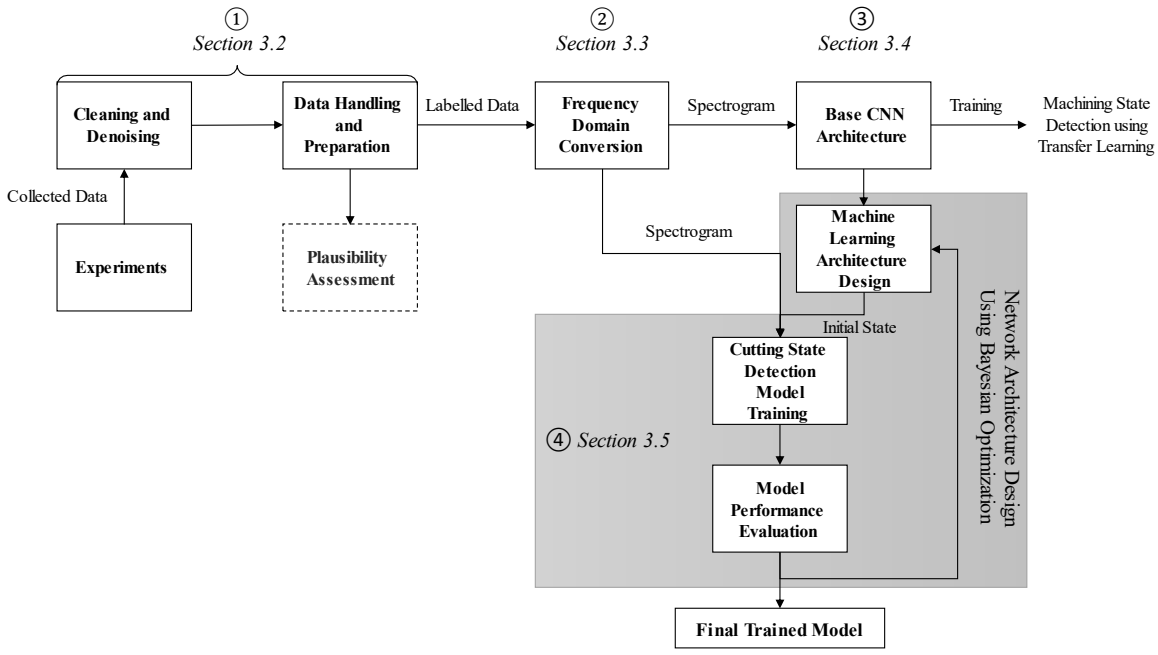

Figure 6 Machine Learning Architecture Design and Algorithm Training Process Overview

In general, there are four steps taken in order to construct a functioning machine learning system:

1) Data Collection, Cleaning, Labelling, and Preparation.
    1.1) Design of a wide variety of cutting tests.
    1.2) Cleaning and Denoising the recorded data.
    1.3) Storing data in a suitable format and manually labeling them based on surface quality and knowledge of machining.
    1.4) Assessing the plausibility of solving the problem using Machine Learning.
2) Feature engineering by Transforming to Frequency domain.
3) Outlining the Base Architecture.
    3.1) Determining the machine learning method of interest based on the characteristic of the problem at hand.
    3.2) Outline the initial input to the Automated Architecture Design Loop.
4) Automated Architecture Design Loop.

4.1) Parameterization of the Machine Learning method's hyper-parameters and define the Optimization Search Space.

4.2) Training the model corresponding to each set of hyper-parameters suggested by the Optimization method.

4.3) Definition of an objective function for the evaluation of the model performance.

The section corresponding to each of the mentioned step is denoted in Figure 6 by a circled number. As illustrated in Figure 6, the architecture design and training processes start with the cutting experiments. After each experiment, the cutting states and stability are assessed manually and stored alongside the signals from sensors and cutting conditions. After having the data denoised, the plausibility of machining state detection using machine learning gets appraised. After finding the cutting state detection problem plausible to be solved using statistical methods, the vibration signals are transferred into the frequency domain. Afterwards, a base Deep Learning network gets trained on the denoised and transformed data. In this step, feature engineering, i.e. setting up the frequency domain conversion parameters, is done. The resultant network is assessed in terms of performance and based on the results. Finally, network architecture gets evolved to become optimal for the purpose of cutting state detection. The optimization is done with the goal of increasing accuracy while minimizing architecture complexity.

In this thesis, a supervised learning approach is adopted for cutting state detection, meaning that a prelabelled set of data, labeled by machining metrics, has been used to train the model to detect different machining states and determine cutting stability.

3.2 Experiment Design, Data Collection, and Plausibility Assessment

The first step towards the fabrication of a data-driven method for chatter detection is collecting cutting data from machining tests. In order to have a robust, generalized, and reliable model, having a diverse set of data collected under different cutting conditions is necessary. Hence, a variety of experiments with five tools, fluted and indexed, on steel AISI 4340 and aluminum 7075, with various tool-paths under different cutting conditions, have been performed. It results into more than 1600 cutting experiments cases. In the designed experiments, vibration signals are recorded to identify the relative tool-workpiece engagement and detecting instability in the cutting process.

Multiple sensors (including microphone, accelerometer, forces, etc.) are used to record signals during the machining process. The experiments are conducted on Quaser UX600 5-axis machining center controlled by Heidenhain TNC 640 CNC as shown in Figure 7.

Figure 7 Experiment Setup

The collected data are labelled as Air Cut, transient state of tool Entrance into the workpiece, Stable Cut, Chatter, and transient state of tool Exiting the workpiece, based on process knowledge and resulting surface quality. An example of each of the mentioned Machining States is shown in

Figure 8. Air Cut describes the state during which the cutter is not in contact with the workpiece. Entrance and Exit describe the transient states during which the cutter is not in its full engagement with the workpiece, and it is getting inside or getting out of the workpiece, respectively. Transient vibrations occur during the entrance and exit transients, with a wide variety of frequency contents, commonly at the same frequency as chatter, due to its similarity to an impact. Hence, there is a need to separate them from the chatter. Finally, Stable Cut and Chatter describe the stability of the cut while the cutter is fully engaged.

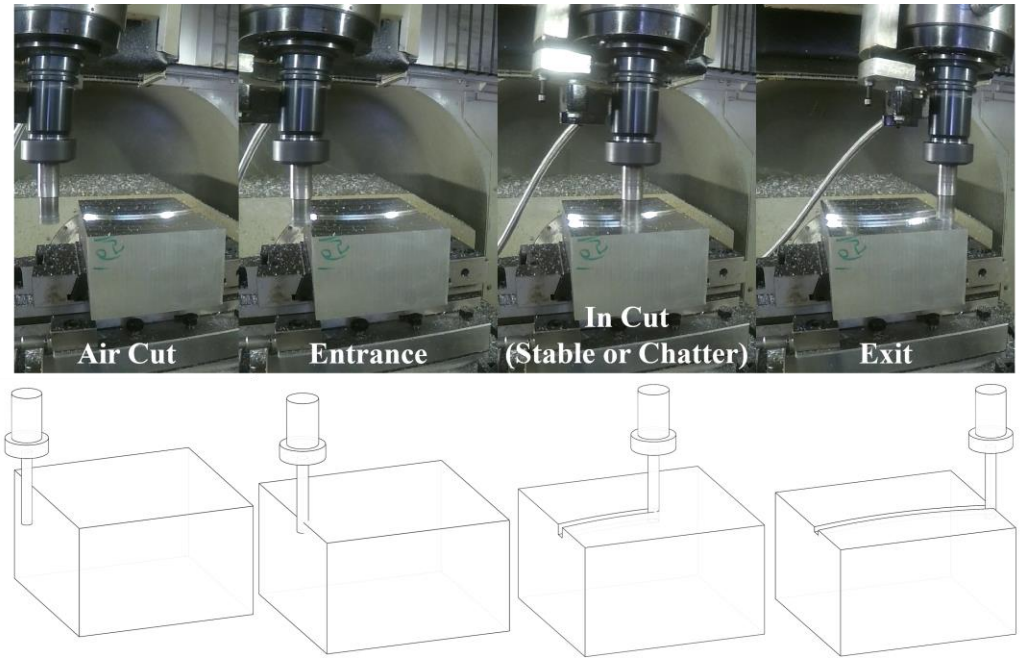

Figure 8 Cutting States Illustration - from left to right respectively, Air Cut, Entrance, In Cut, and Exit

For accuracy in determining each of the cutting states mentioned in the last paragraph, multiple factors are taken into account during the phase of manually labelling the experiment data. The machine's feed rate and the G-Code are used to evaluate the precise relative tool-workpiece relative position. Furthermore, since chatter leaves vibration marks on the surface, the surface quality after the cut has been used as the criteria for cutting stability assessment, i.e. in order to determine whether it is chattering or not. The mentioned states and stability conditions have been

used during the training and automatic design process, as well as the test and performance evaluation process.

The cutting tests have been partitioned into three subsets of Training, Validation, and Test. The mentioned subsets contain 76, 12, and 12 percent of the whole data, respectively. The training and validation subsets are similar in terms of cutting conditions. As shown in Figure 4, the training set has been used to tweak the model parameters during the training process. The test data consists of more than 30% of unseen cutting conditions to ensure that the reported performance is generalized and realistic and not a result of overfitting. Overfitting a machine learning model means having the model extrapolated to the part of the input, which does not contribute to the results. Overfitting commonly happens when the model attempts to get fitted to residual variation and noises unintentionally and makes predictions based on them. There are methods and techniques to avoid overfitting, which are extensively practiced in this thesis. It is further explained in this chapter and through the thesis.

Moreover, the Test set remained untouched and merely used to report the model's performance and its subcomponents. In other words, the test subset has not been contributed to the design, training, or tuning of the cutting state detection system or the models, and it has been used only for the evaluation of the system performance. A summary of how the experiments have been used during each process is presented in Figure 9.

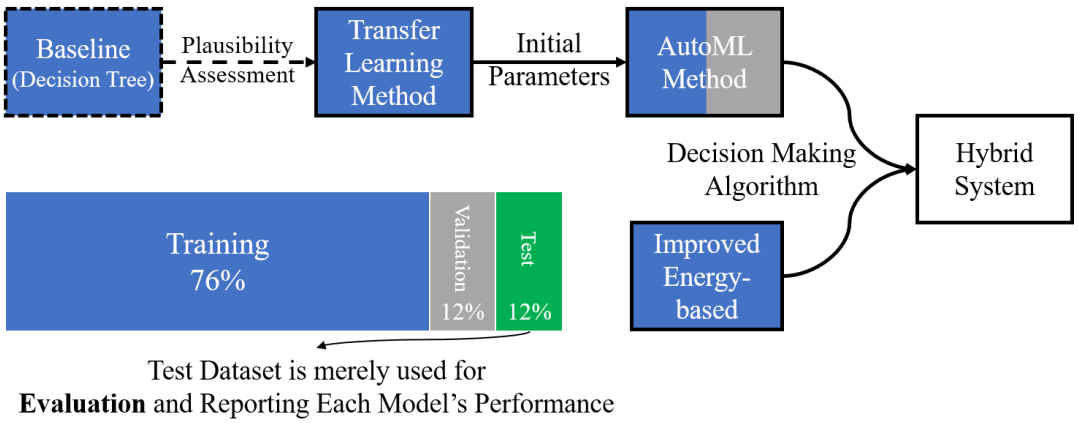

Figure 9 Experiments' Data Partitioning and Utilization Summary

3.2.1 Cleaning, Denoising, and Handling Data using Comprehensive Industrial Process Monitoring Data Structure

High-frequency reading from multiple sensors results in a high volume of data; hence, managing and handling the computer resources efficiently is crucial during the whole process, especially the training process. Moreover, it is necessary to have a data type designed to accommodate the multi-sensor, time-series measurements' requirements. This data structure's main objectives are to speed up the training process and increase the robustness, scalability, and versatility of the training pipeline.

In Figure 10, in the designed data structure, each entry has three primary attributes: index, measurement source (e.g. microphone, and accelerometer), and label (e.g. air cut, entrance, stable, chatter, and exit) associated with each part of the signal. The three mentioned attributes are a unique identifier of the entry.

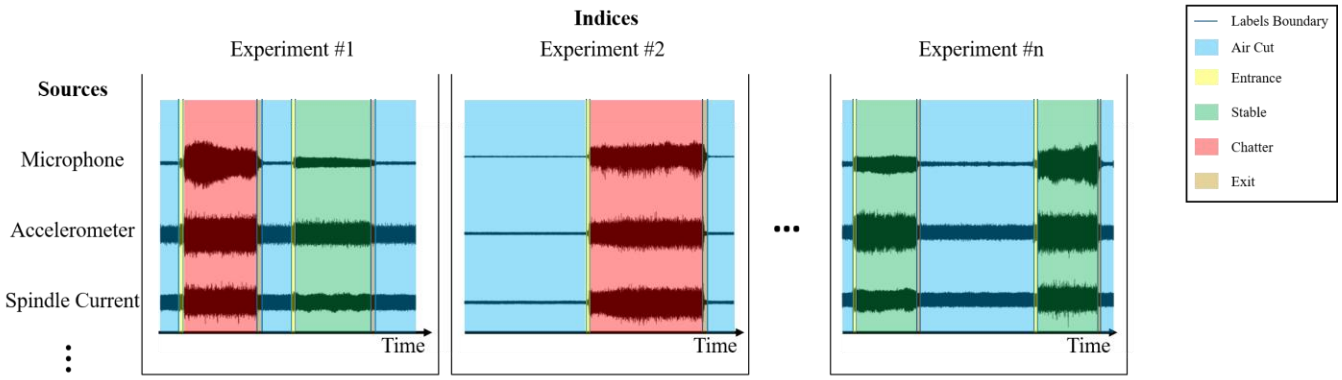

Figure 10 Overview of Comprehensive Industrial Process Monitoring

Each set of synchronized signals is partitioned into smaller pieces of signal to be used in the training process, as illustrated in Figure 11. It provides appropriately sized training instances. Moreover, the designed data structure offers embedded data augmentation by breaking down each index into multiple sets of smaller pieces of signal with a time delay between them, similar to Rebai et al.[24] suggested in their paper. A scheme of the embedded data augmentation has been depicted in Figure 11, where the data between each consecutive pair of red or yellow dashed lines counts as a cutting instance.

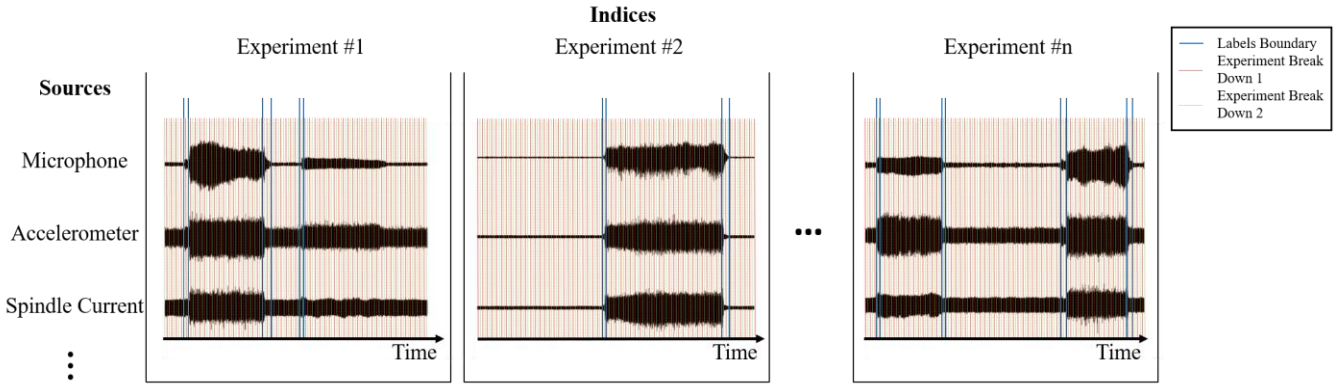

Figure 11 Breaking down the data and Embedded Data Augmentation

The designed data structure has filtering, and feature extraction embedded, making real-time filtering and transformation easier. The designed data structure stores data in RAM in the form of

pointers to the permanent memory, which results in having each data point taking no more than a few bytes on RAM. The chunks of experiment measurement data could be summoned from permanent memory in a queue, with a shuffleable order for randomizing and avoiding overfitting, or it could be fetched by having its set of unique identifiers. Given that reading from the permanent memory into RAM is the bottleneck, the designed data structure minimizes the permanent memory queries by predicting the next sets of data to be summoned and read them all into RAM once. Moreover, the data structure can read and process the data in parallel to reduce the computation time by utilizing full computational capacity and multithreading. Moreover, a three-dimensional sorted lookup table made it possible to access the sets of data based on each combination of identifiers (i.e. combination of Experiment Number, Source, and Label) in $O(\log(n))$ plus the reading time. Each index is accompanied by a properties table that contains the other attributes of each experiment, such as the cutting condition and the tool properties in the use case of this thesis.

In this thesis, data acquisition is made through a custom-written LabVIEW application designed for this purpose. The data is stored and organized in the described data structure. The label assigned to each entry is its machining state and determined based on the last section's mentioned method. Both denoising and removing the signal's drift happens through the data structure, as well. For denoising, a high-pass filter at 4 Hz is used to eradicate the low-frequency noises. In order to get the signal drift and DC offset eliminated from the input signal, the DC blocker proposed by M. Nezami [25] has been used. For the purpose of training, the cleaning and denoising process happened after data collection. However, during online detection, it happens online during the data collection and through the custom data collection application designed for this purpose.

3.2.2 Plausibility Assessment and Setting a baseline

A baseline is a plausibility assessment method, commonly using heuristics, summary statistics, and randomness measurement. Setting a baseline is usually done using a classic machine learning method, in contrast to deep learning methods. In machine learning problems, it is substantial to set a baseline to assess the plausibility of solving the problem using machine learning before getting started with deep learning. The outcome of this section is a machine learning model fully capable of chatter detection. Nevertheless, the baseline will not be used in the final system, and it is merely for feasibility assessment. In this section, the method of the baseline calculation is explained.

As shown in Figure 12, the first step is to break down the measurements into smaller pieces, using the comprehensive industrial process monitoring data structure. For training, 27000 cutting instances labelled to five matching states, each consists of 0.2 seconds of the microphone vibration signal, are generated.

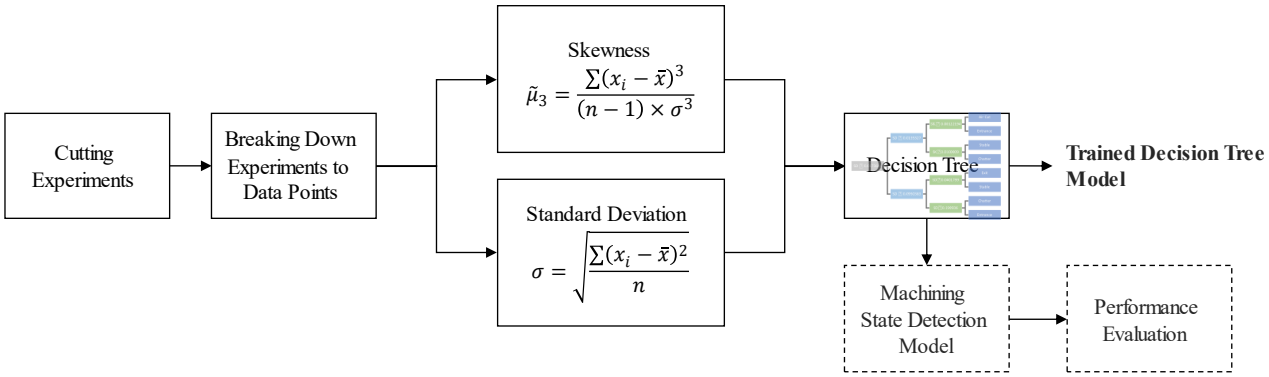

Figure 12 Setting the Baseline Procedure

The considered statistical measures that have been used for this cutting state classification problem are standard deviation and Skewness. Standard deviation is a measure of dispersion, and it is equal to the square root of the variance. A higher standard deviation indicated that the vibration signal's

magnitude differs from the average magnitude of the signal. The standard deviation of $x_1, x_2, \dots, x_n$, where each x_i is the magnitude of the signal in an instance of discrete-time is determined by,

$$\boldsymbol{\sigma} = \sqrt{\frac{\sum(\boldsymbol{x_i} - \bar{\boldsymbol{x}})^2}{\boldsymbol{n}}} \tag{31}$$

Where $\bar{x}$ is the mean of the x_i values.

Furthermore, skewness is a measure of asymmetry in the distribution of a set of data. Applying Skewness to time-domain signal measures the similarity of the measured signal amplitude to the normal distribution. In other words, skewness for a normally distributed set of values is equal to zero. For $x_1, x_2, \dots, x_n$, where each x_i is the magnitude of the signal in an instance of discrete-time is determined by,

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{3}} = \frac{\sum(\boldsymbol{x_i} - \bar{\boldsymbol{x}})^{\boldsymbol{3}}}{(\boldsymbol{n} - \boldsymbol{1}) \times \boldsymbol{\sigma}^{\boldsymbol{3}}} \tag{32}$$

A decision tree is used as a machine learning model to classify the cutting instances to corresponding machining states. A decision tree is a straightforward machine learning method that determines a set of linear decision boundaries to perform classification, in the case of this work differentiating among the machining states. Each decision boundary is a threshold that assigns a cutting state to each measurement based on the values associated with it.

For training the model, a 10-fold leave one out cross-validation scheme has been used to protect the model against overfitting, which means that the data points have been partitioned into ten disjoint subsets. For each subset l_i ($i = 1, 2, \dots, 10$), the model has been trained on the existing data

points in the other nine subsets, all together, and the model performance has been assessed based on the left-out subset (i.e. subset x). The rest of the subsets will go through the same procedure, and the final error will be calculated based on the average error of all ten subsets [26].

Given the lower count of the data labelled as entrance and exit, in the loss function, the misclassification cost of the mentioned labels is ten times higher than any other misclassification.

The model's accuracy is assessed based on its performance in detecting cutting states of the test set. The results of the method are presented in section 5.2. If the baseline shows promising performance, it is rational to consider using a more advanced machine learning method to solve the problem. In the case of this thesis, it is decided to use a deep convolutional neural network.

3.3 Frequency Domain Conversion and Feature Extraction

The signal’s frequency domain characteristics are an indicator of the cutting state. Whether tooth passing frequency is being observed, in the frequency domain or not, could be an indicator of the tool being in or out of the cut. Moreover, in the case of chatter vibration, resonance peaks and associated harmonics are observed in the frequency domain.

The frequency-domain signal is a good indicator of the cutting state. In this thesis, frequency domain features have been utilized to differentiate between different states of cut and detect the stability of cut.

For the observation of the changes in the amplitude trends in frequency-domain features of the signal over time, a short-time Fourier transform (STFT) has been applied to the signal [27]. STFT calculates the Fourier transform during the time. STFT is commonly used to assess the frequency content of a nonstationary signal over time. The purpose is to use frequency-domain features to

differentiate between different states of cut and detect chatter. The machine learning network takes a duration of 0.14 seconds of vibration data as its input and extracts its frequency domain features using STFT.

To calculate STFT the signal gets windowed, with overlap. For 0.14 seconds of vibration data, equivalent to 7000 sampling point with a sampling frequency of $f_s = 50kHz$, the windowing process is shown in Figure 13. Windows are overlapping over a length L_o to compensate the signal attenuation at the window edges. The overlap provides more continuous frequency domain data during the time. The window size is $L_w = 800$ and the overlap is $L_o = 680$ in this work. The number of windows for each milling case of n samples is evaluated as $\mathrm{L} = \left\lfloor \frac{n-L_o}{L_w-L_o} \right\rfloor$.

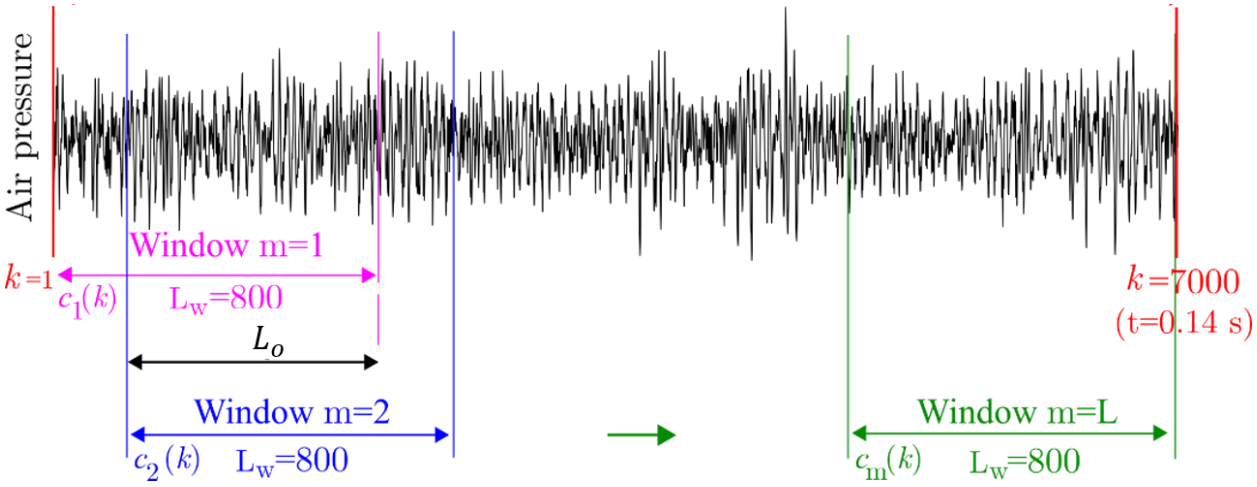

Figure 13 Windowing of Vibration Signal for Calculating STFT

As shown in Figure 15, the STFT of a signal is determinable by applying a Hanning window [28] to each windowed data of length L_W and calculating the discrete Fourier transform (DFT) of the resultant signal. The STFT windowing method results in frequency-domain signal through discrete-time domain, covering the frequency components up to Nyquist frequency.

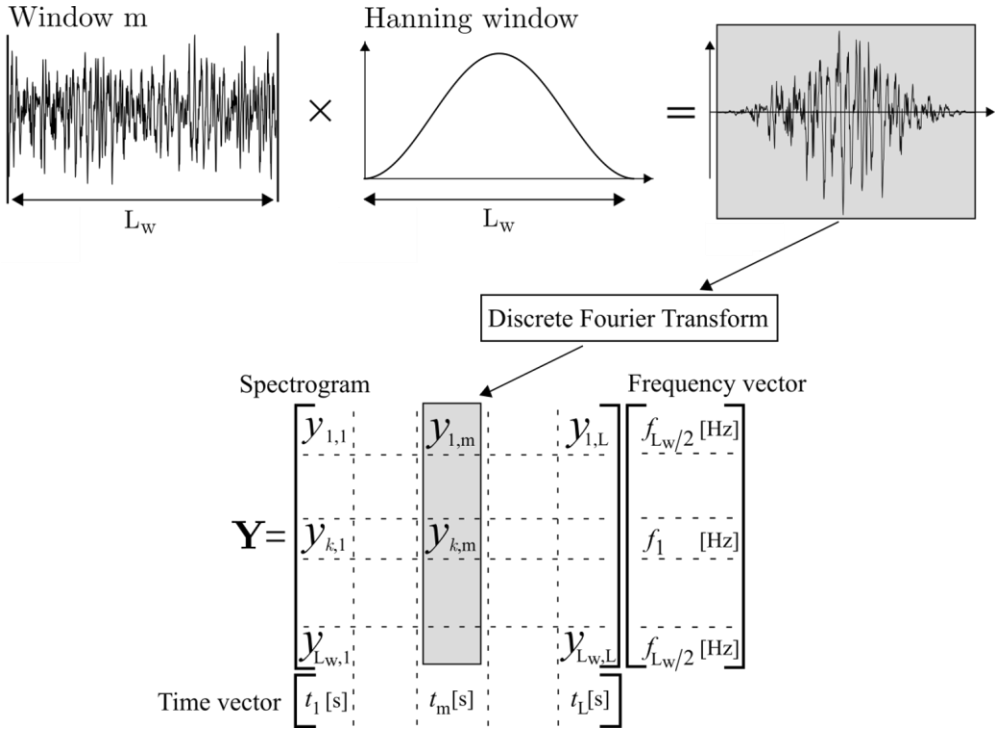

Figure 14 STFT for each given Instance of Vibration Signal

For a given vibration signal of $c_m(kT)$ in discrete time domain, where $k = 1,2, \dots, n$, each element of the STFT matrix is computed by,

$$y_{k,m} = \sum_{i=1}^{L_w} c_m(iT)\, g(i) e^{-j2\pi\left(\frac{L_w}{2}-k\right)\frac{(i-1)}{L_w}} \quad \text{where } 1 \le k \le Lw \text{ and } 1 \le m \le L \tag{33}$$

Where g is the window function of the length L_w. The window function in the case of this work is Hann window [28] determined by,

$$g(i) = \frac{1}{2}\left[1 - cos\left(\frac{2\pi(i-1)}{L_w}\right)\right] \quad \text{where } 1 \le i \le L_w \tag{34}$$

The magnitude squared of STFT matrix elements also referred to as spectrogram, is used to detect chatter based upon. In Figure 15, the process of calculating the spectrogram is shown. The resultant STFT matrix for each 0.14 seconds of machining instance in $f_s = 50kHZ$ is 400 by 52.

Additionally, the mapping of the $|\mathrm{STFT}|^2$ to the visual representation of the STFT of a signal is shown. The shown visual representation is used repeatably in this work. Each column of the STFT matrix corresponds to a time instance, and each row corresponds to a frequency is illustrated as a colour coded two-dimensional matrix.

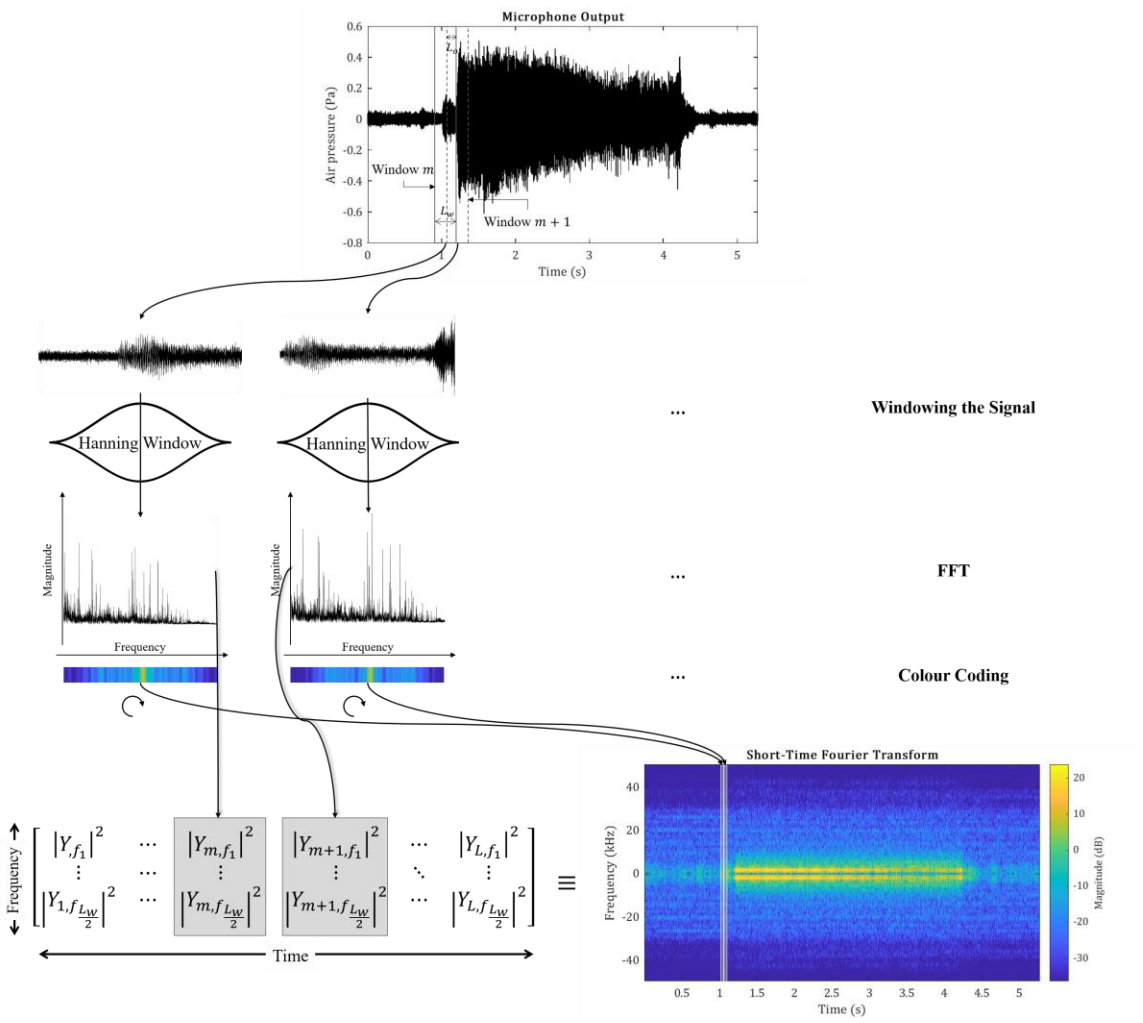

Figure 15 Short-Time Fourier Transform's Matrix Visual Representation Example

The windowing used in this work satisfies constant overlap-add (COLA) constraints [29]. Finally, the resultant STFT matrix is converted to a decibel scale and linearly scaled to the range of zero to one to avoid the bias effect of the magnitude and avoid overfitting. Since amplitudes are

symmetric from across the Nyquist frequency, half of the spectrogram is kept resulting in the spectrogram on the right-hand side. Figure 16 shows a scaled spectrogram with only the positive frequency range from an instance of stable cut. The matrix shows the intensity of the vibrations occurring at each frequency, leading to the detection of forced vibrations at the harmonics of tooth frequency and chatter close to one of the structural modes.

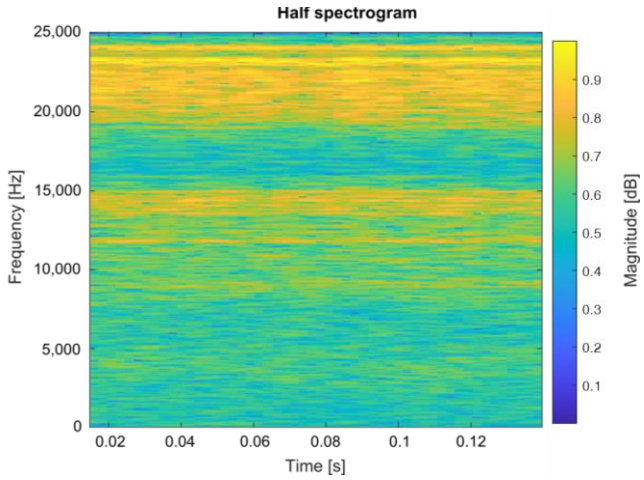

Figure 16 STFT Example

3.4 Machining State Detection using Transfer Learning

In this section, the transfer learning method used for machining state detection and its components are explained. The components are used further in the next section.

The base architecture for machining state detection is a modified version of a Convolutional Neural Network (CNN) called ImageNet [30]. The proposed method takes the frequency domain signal generated by section 3.3's method as input and outputs the probability of each state of machining. The work explained in this section has four importance. First, transfer learning provides fast and robustness training due to being pre-trained. Hence, it provides a reliable cutting state detection system fully capable of detecting machining states independently. Secondly, the transfer learning architecture is used during the process of feature engineering, i.e. finding the right frequency

domain parameters. Third, the resultant trained model's weights and biases are scaled and used as the initial weights and biases for training of the custom designed architecture. Most importantly, a modified version of this work is used as the initial input to the architecture design loop.

A convolutional neural network is a deep learning architecture consisting of two major parts: a set of convolutional blocks and a neural network. Each convolutional block consists of a two-dimensional discrete convolution layer followed by a number of other arithmetic layers. The first block has the network's STFT input as its input. The next blocks each have their previous block's output as their input and performs arithmetic processes to generate output in a chain. The convolutional layers extract the form attributes of the input STFT and decrease its size. The output of the convolutional blocks gets flattened and goes into an artificial neural network. Using Artificial Neural Networks (ANN), an equation gets fit to the output of the convolutional blocks. The output of this equation for any given vibration signal input is the probability of the signal being associated with each of the machining states.

To sum up, CNN is a set of weights and biases, forming an equation. The equation gets fit to the machining state detection problem.

Further through this section, the deep leaning training pipeline and its building blocks for the base architecture are explained. These blocks are used further in the next sections, as well.

3.4.1 Deep Learning Pipeline Overview

A pipeline is a sequential set of data flow rules among functions and models. There are training and execution pipelines designed in this thesis. The final resultant model of the close loop training pipeline is used in the chatter detection pipeline. The training pipeline is designed to train a model

to detect machining states based on vibration signals. The pipeline for training the base network is shown in Figure 17.

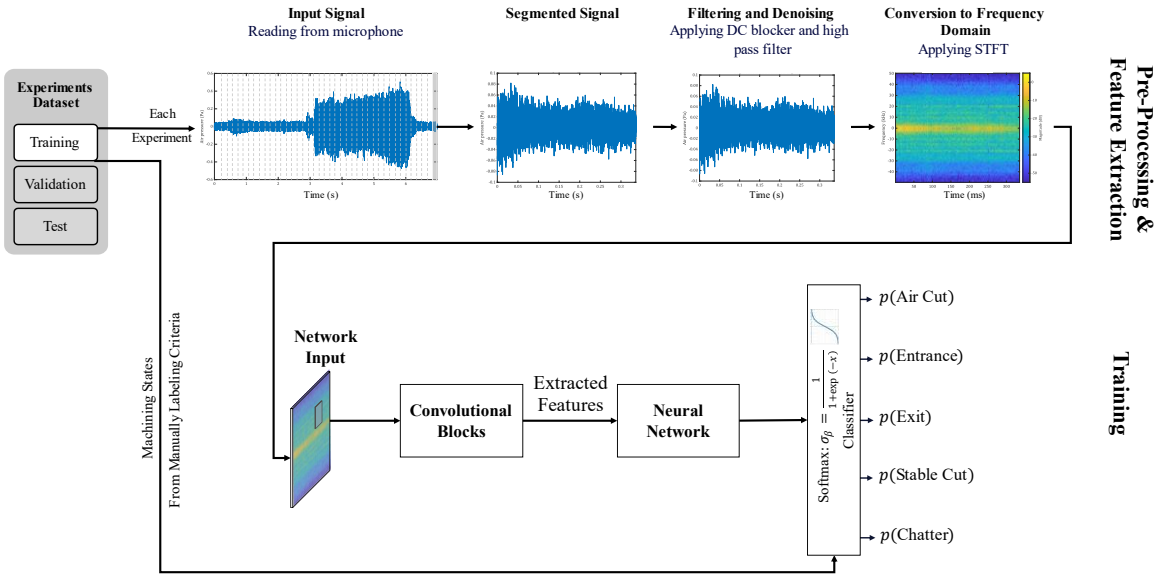

Figure 17 Deep Learning Training Pipeline Overview

The storage, segmentation, filtering, and transformation to frequency domain steps are elaborated so far. This section's focus is on the deep learning part of the training pipeline, i.e. convolutional blocks and ANN.

To have a functional machine learning model capable of machining state detection, it is crucial to construct a suitable learning architecture. Hence, a well-known deep learning architecture from ImageNet Classification with Deep Convolutional Neural Networks [30] has been adapted to machining state detection and used as the base architecture. In a process known as transfer learning, the pre-trained network is retrained in order to detect machining state and chatter stability. ImageNet is a relatively small/non-complex deep learning architecture. During the training process, the weights and biases are adjusted to make the network equation serve the goal of machining state detection.

After training the network, the trained model will be used for machining state detection. The input is generated online, by denoising the vibration signal and calculating the spectrogram. Afterwards, the model will be applied to the data in order to calculate the probability of the current vibration being associated with each of the five machining states. The state with the highest probability will be detected as the current state.

The components of the mentioned deep learning architecture are convolution layers, arithmetic and activation functions (e.g. ReLU, Normalization, and Max pooling), and neural network layers. Further through these subsections, the building blocks of the deep learning method are explained, and the notation presented above is elaborated.

3.4.2 Convolutional Blocks

The input to deep learning architecture goes through a set of convolutional blocks. Each block starts with a convolutional layer followed by a set of arithmetic functions. Each convolutional block contributes to the discovery and isolation of the relative changes in the frequency domain signal amplitude during the time, which contributes to chatter or other machining states. A sample convolutional block with the notation convention, followed in this thesis, is shown in Figure 18,

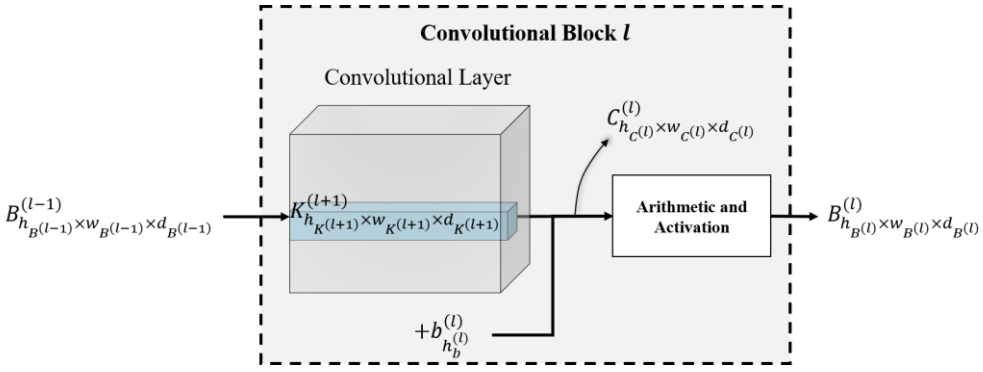

Figure 18 A Generic Convolutional Block's Notation

In the figure, $C^{(l)}_{h_{C^{(l)}} \times w_{C^{(l)}} \times d_{C^{(l)}}}$ and $B^{(l)}_{h_{B^{(l)}} \times w_{B^{(l)}} \times d_{B^{(l)}}}$ are the output matrices of the convolutional layer of the l-th convolutional block and the output of the l-th convolutional block, respectively. $B^{(l)}_{h_{B^{(l)}} \times w_{B^{(l)}} \times d_{B^{(l)}}}$ is the output of a series of arithmetic layers, such as activation, normalization, and max pooling, for the input $C^{(l)g}_{h_{C^{(l)}} \times w_{C^{(l)}} \times d_{C^{(l)}}}$. Moreover, $K^{(l+1)}_{h_{K^{(l+1)}} \times w_{K^{(l+1)}} \times d_{K^{(l+1)}}}$ is the convolution kernel of the $l+1$-th layer, which will be applied to its previous block's convolutional layer. Each layer l has a bias term of $b^{(l)}_{h_b^{(l)}}$. The equation of the l-th block is,

$$C^{(l)}_{h_{C^{(l)}} \times w_{C^{(l)}} \times d_{C^{(l)}}} = \tilde{B}^{(l-1)}_{h_{B^{(l-1)}} \times w_{B^{(l-1)}} \times d_{B^{(l-1)}}} * K^{(l+1)}_{h_{K^{(l+1)}} \times w_{K^{(l+1)}} \times d_{K^{(l+1)}}} + b^{(l)}_{h_b^{(l)}} \quad (35)$$

Where the asterisk is the convolution operator and $\tilde{B}^{(l-1)}$ is the $B^{(l)}$ with the padding applied to it. The arithmetic and activation functions get applied to the value of $C^{(l)}$ to generate the output of the block ($B^{(l)}$).

The kernel ($K^{(l)}$) and the bias ($b^{(l)}_{h_b^{(l)}}$) term of the convolutional layer are the learnable parameters that are determined during the training process.

In the next subsections, the convolution layer and arithmetic operations, rectified linear unit, local response normalization, and max pooling, occurring in the convolutional blocks are detailed.

3.4.2.1 Two-dimensional Discrete Convolution Layer

The convolution layer is the most important part of the convolutional block, as it is the only part of each convolutional block that gets tweaked during the training process. The two-dimensional discrete convolution layer consists of a set of two-dimensional matrices (which is denoted as

$K^{(l)}_{h_{K^{(l)}} \times w_{K^{(l)}} \times d_{K^{(l)}}}$ as a three-dimensional matrix) which will be determined during the training process. It gets convolved with its input ($B^{(l-1)}_{h_{B^{(l)}} \times w_{B^{(l)}} \times d_{B^{(l)}}}$) to generate an output ($C^{(l)}_{h_{C^{(l)}} \times w_{C^{(l)}} \times d_{C^{(l)}}}$). A two-dimensional convolution reduces the input size and preserves and frames the frequencies and the change in frequencies over time, the STFT parts, which contributes the most to the cutting states. In other words, during the training process, discrete convolution kernels form to highlight and pick the parts of the frequency domain signal, which play a role in causing chatter or other states of machining. As mentioned, each convolution block starts with a convolution layer.

The two-dimensional convolution layer calculates the discrete convolution between the input and the convolution receptive field by iterating the convolution kernel matrix over the layer's input and evaluating the dot product of the input and the kernel at each point [17]. A two-dimensional convolution layer has a bias term that adds to each element after the convolution is applied.

In this thesis, the first and third convolution operations are applied to their entire inputs, while the rest of the convolutions are two-dimensional grouped convolution. Two-dimensional grouped convolution divides its input into groups, in this thesis, two groups, and applies sliding convolution on each of them. It helps with decreasing the number of learnable parameters.

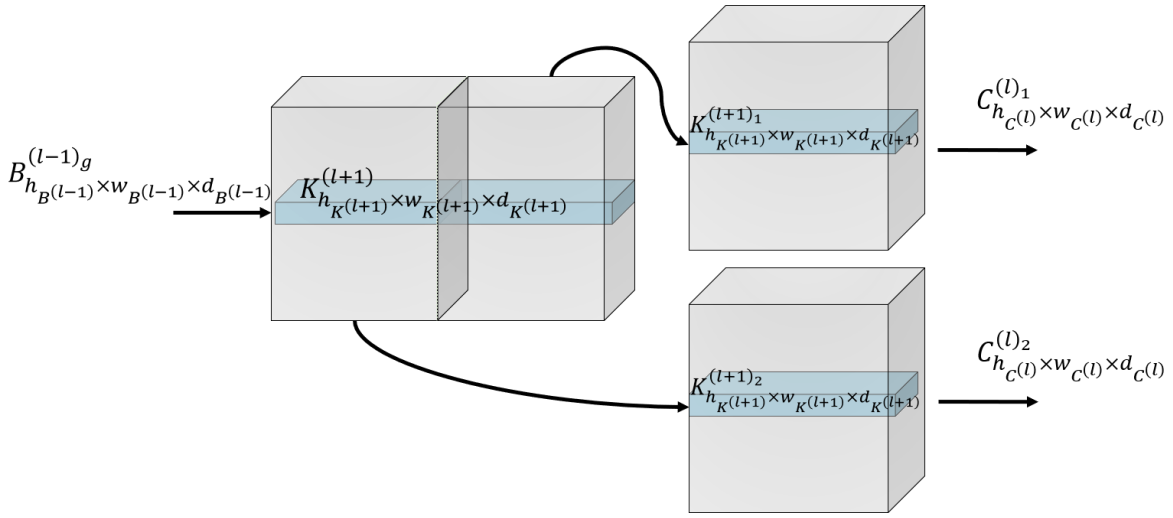

Figure 19 Illustration of group convolution function for a two-group, group convolution

Two-dimensional grouped convolution divides its input into groups, in the case of this thesis, two groups, and applies sliding convolution on each of them. Two groups grouped convolution results in having each kernel being convolved with a number of feature maps obtained by the previous layers. It results in having less redundancy, in term of similarity of the weights and biases or the block's outcome, for kernels.

Two important terms to define when it comes to Convolutions are stride and padding. Stride is the step size of the convolution window and denoted by $k_s^{(l)}$. Padding is the rows and columns added to the input's margins to give the kernel more space to cover the whole input. An example of an arbitrary 3×3 convolution kernel being applied to the input of the CNN network with zero paddings and a stride of $k_s = 4$ is illustrated in Figure 20.

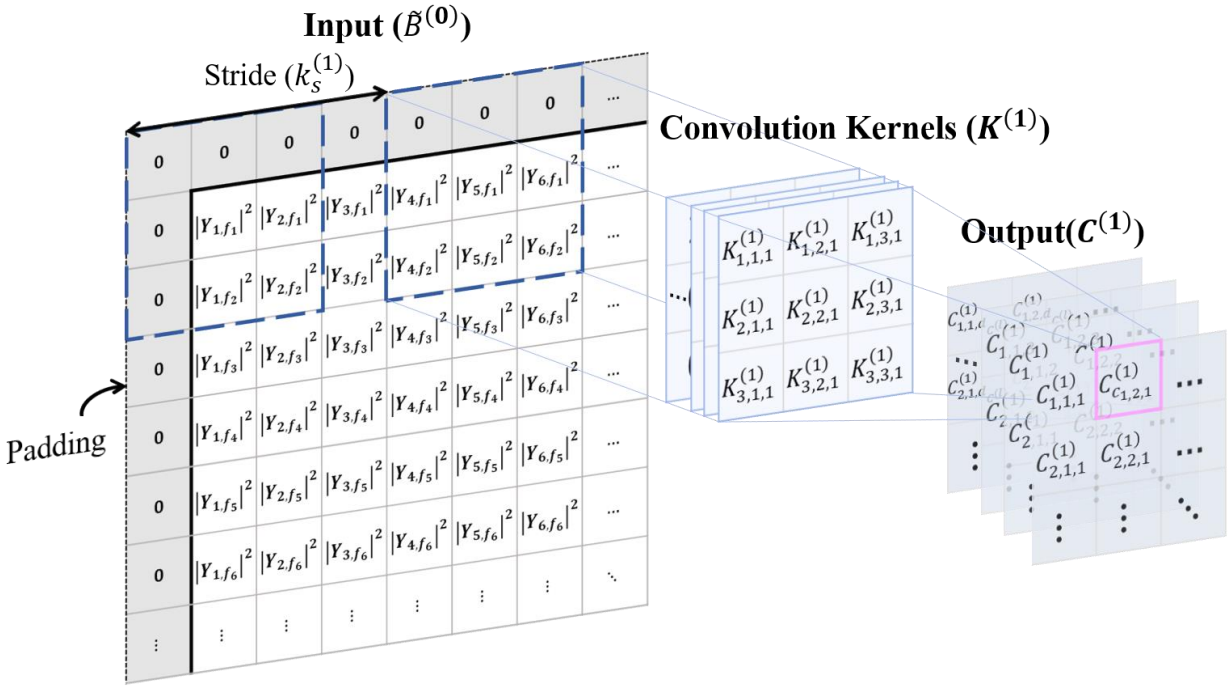

Figure 20 Discrete Convolution Illustration with an Arbitrary Kernel Example

The output of the convolution layer l is given by $C^{(l)} = \tilde{B}^{(l-1)} * K^{(l)} + b_K^{(l)}$, where $C^{(l)}$ is the output of convolutional layer l, also referred to as the activations, $B^{(l-1)}$ is the output matrix of layer $l-1$, $\tilde{B}^{(l-1)}$ is the padded $B^{(l-1)}$, $K^{(l)}$ is the convolution kernel of layer l and $b_K^{(l)}$ is the bias term of the same kernel. In this convention $l = 0$ denotes the input. For a two dimensional convolution, each element of $C_{i,j,k}^{(l)}$ ($i = 1, 2, \dots, h_{C^{(l)}}$, $j = 1, 2, \dots, w_{C^{(l)}}$, $k = 1, 2, \dots, d_{C^{(l)}}$) matrix is given by,

$$\boldsymbol{C_{i,j,k}^{(l)}} = \sum_{\boldsymbol{a=1}}^{h_{K^{(l)}}} \sum_{\boldsymbol{b=1}}^{w_{K^{(l)}}} \boldsymbol{\tilde{B}^{(l-1)}_{(i-1)\times k_s^{(l)}+a+1,(j-1)\times k_s^{(l)}+b+1,k}} \times \boldsymbol{K_{a,b,k}^{(l)}} + \boldsymbol{b_k^{(l)}} \qquad (36)$$

During the training process, the value of each convolution kernel and its biases change in a way to make the whole CNN network capable of highlighting the critical features of the spectrogram, i.e. the relative magnitude of the signal in each frequency, which contributes to chatter as well as other

machining states. From a conceptual point of view, convolution kernels determine patterns and changes in the amplitude of the input signal's different frequencies during the time.

Below a sample STFT of a stable milling process with the spindle speed of $2400[\mathrm{rev}/min]$ and a four inserted tool with runout is shown to illustrate the effect of a sample horizontal line detector kernel and showcase the potential of convolution kernels.

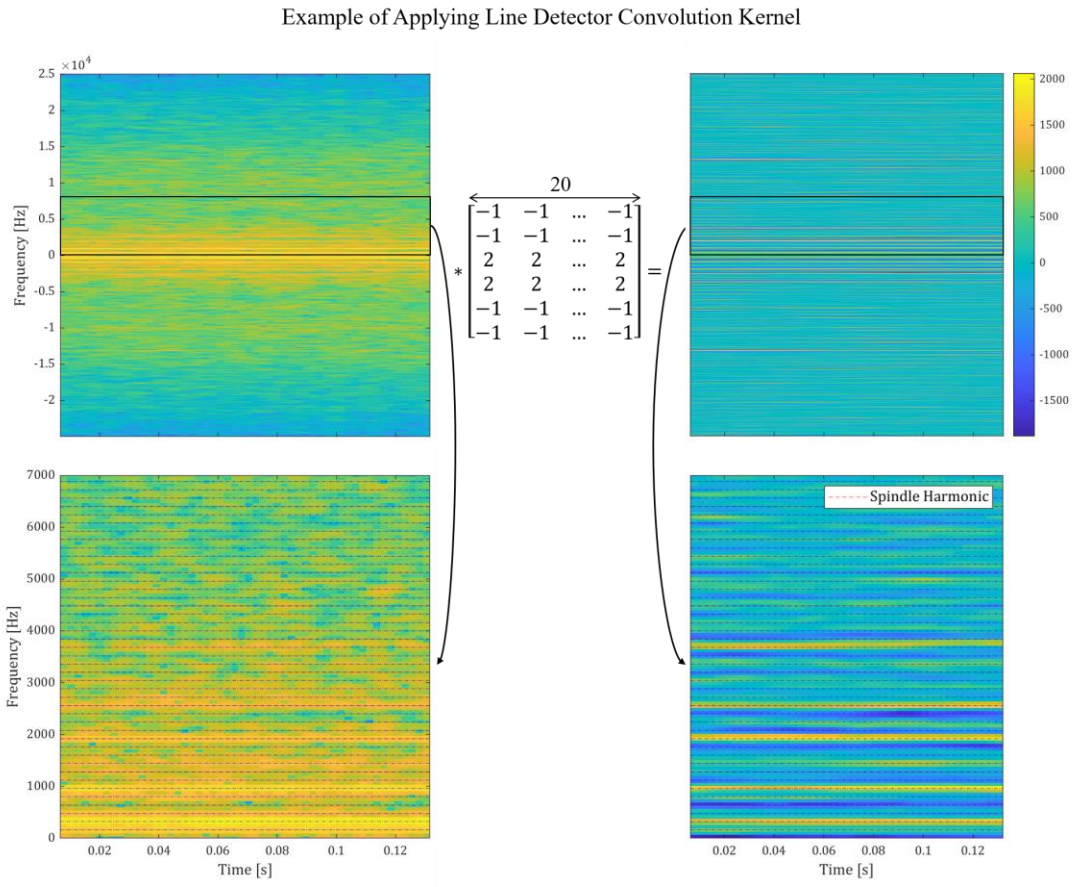

Figure 21 Applying a Line Detector Convolution Kernel on STFT Data Example

3.4.2.2 Rectified Linear Unit (ReLU)

A Rectified Linear Unit is an activation function. It introduces non-linearity to the equations while it does not saturate. It preserves nonnegative input values and replaces the negative values with zero. The Rectified Linear Unit applies the below function to each element of its input matrix,

$$\mathbf{ReLU}(\boldsymbol{x}) = \mathbf{max}(\mathbf{0}, \boldsymbol{x}) \tag{37}$$

ReLU highlights the most critical parts of the frequency domain signal for machining state detection.

The effect of applying the ReLU function on the data of Figure 21 is shown below. As illustrated, the horizontal lines, which indicate the consistent frequencies during the time, appear to get more prominent after applying ReLU.

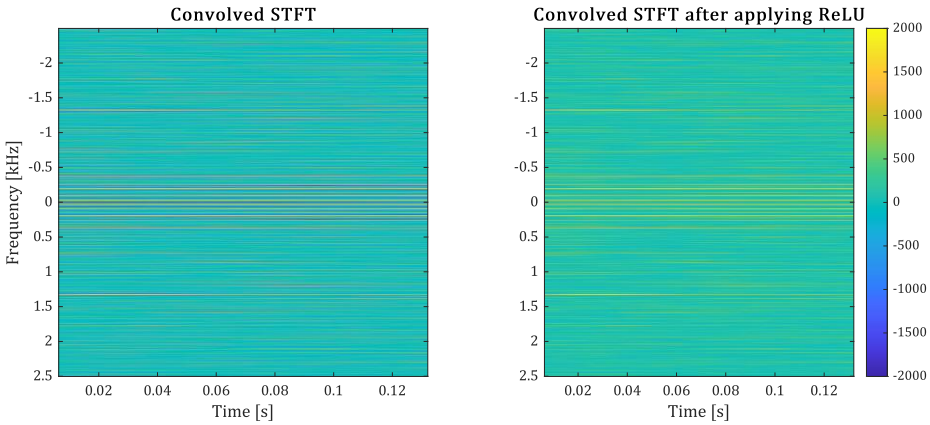

Figure 22 Example of Applying ReLU to Convolved STFT Matrix

3.4.2.3 Local Response Normalization

Each convolution layer could consist of multiple convolution kernels. Local Response Normalization layer performs lateral inhibition by having each high amplitude output result of each convolution kernel, affecting the adjacent kernels output. In other words, it carries out the changes in the amplitude picked up by one convolution kernel to the neighbour channels.

Local Response Normalization performs a channel-wise local response normalization by replacing each element with a normalized value obtained based on its neighbours [30]. Normalizing in the local regions is a kind of Surround Inhibition [31].

For an arbitrary input of $\boldsymbol{C_{i,j,k}^{(l)}}$ element in the kernel k, at position (i,j), the normalized response given by,

$$\boldsymbol{CNorm\left(C_{i,j,k}^{(l)}\right) = \frac{C_{i,j,k}^{(l)}}{\left(k + \alpha \sum_{x=\max\left(1,k-\frac{n-1}{2}\right)}^{\min\left(d_{C^{(l)}},k+\frac{n-1}{2}\right)} \left(C_{i,j,x}^{(l)}\right)^2\right)^{\beta}}} \quad (38)$$

where k, n, α, and β are hyper-parameters whose values are determined using a validation set; In the case of this thesis, $k = 2$, $n = 5$, $\alpha = 10^{-4}$, and $\beta = 0.75$. as proposed by A.Krizhevsky et al. [30].

To illustrate the concept, the mechanism of its implementation for a window of size five ($n = 5$) is shown in Figure 23. It is shown how each element of each matrix, highlighted in red, is replaced with the value of all of the highlighted elements, red and blue, going through equation (38).

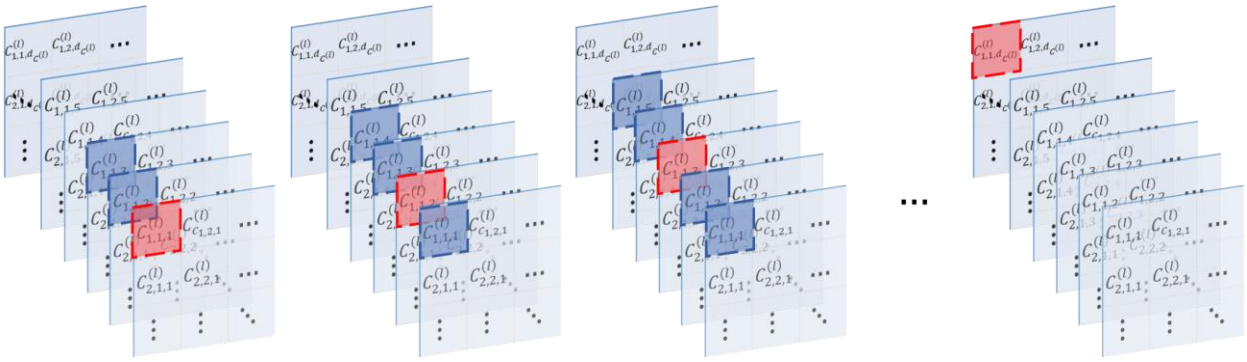

Figure 23 Local Response Normalization implementation illustration

3.4.2.4 Max Pooling

Max pooling is a method of down-sampling. The Max pooling process slides a window over its input matrix sub-regions and takes the maximum value in each window as each subregion representative.

Similar to Rectified Linear Function, max pooling has stride and padding and it puts emphasis on the most matched inputs. Moreover, it eliminates the elements that do not match the amplitude changes according to the convolution kernel. Take the first convolution block as an example; if a max-pooling is added immediately after the convolution layer, each convolution kernel existing in the convolution layer marks the frequency domain signal elements that match the convolution kernels changes of amplitude in different frequencies and during time by showing a high value in the corresponding element of the output matrix. Afterwards, if a max-pooling is added, it will down-sample each convolution kernel's output by preserving the amplitudes that made a better match by eliminating the rest of the amplitudes.

Max pooling on $C^{(1)}$ in the first block of the base architecture, with a stride of two, is shown in Figure 24 as an example.

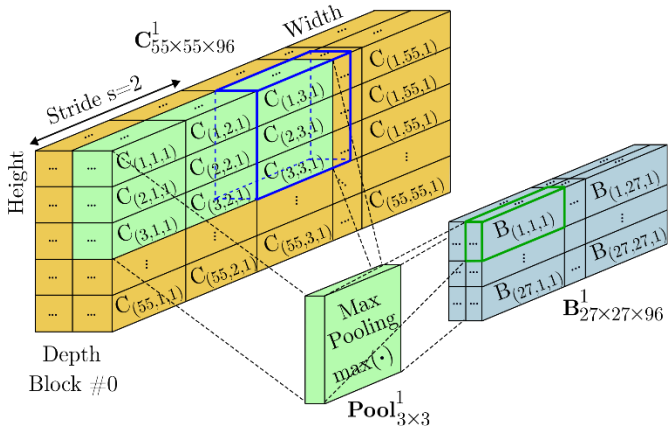

Figure 24 Max pooling Operation Example

The convolved STFT after applying ReLU, as shown in Figure 22, is depicted as an example of the max-pooling function with size of 3 and a stride of 2. In the mentioned figure, max pooling output (on the right) highlights the horizontal lines which were discovered by the convolution kernel and reinforced by ReLU.

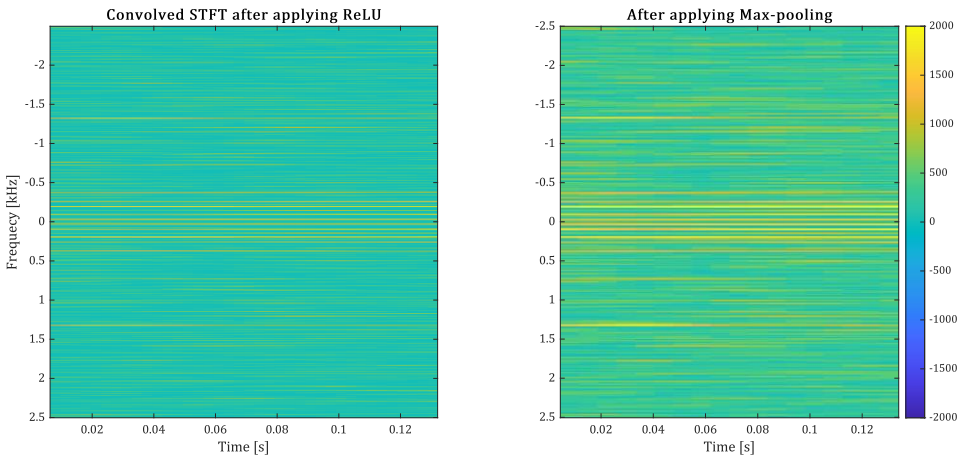

Figure 25 Example of Max-pooling and applying it to a Convolved STFT after ReLU function

3.4.3 Artificial Neural Network Components

The output of the last convolution block is a set of matrices. These matrices are convolved and manipulated from the frequency domain input signal. The convolution layers are supposed to get trained to pick up the frequency domain measurement parts that contribute the most to the cutting states and discard the less/non-important parts.

The flattened output of the last layer of convolutional blocks goes into an artificial neural network (ANN) to be classified into the sought states, i.e. the machining states. The ANN takes each element of the matrices from convolutional layers as one input variable and fits an equation to all the input variables. The output of this fitted algorithm will be a set of five numbers, each corresponds to the probability of one of the five machining states

An ANN is a set of interconnected artificial neurons. Neurons are variables, each capable of storing a number. The interconnections have numbers associated with them, as well. The Interconnection numbers are called weights.

There are various types of ANNs, yet the most commonly used type, due to its wide range of capabilities, is feedforward ANN. Feedforward ANN is an ANN wherein each neuron belongs to

one layer. The layers in feedforward ANN are ordinal, and each neuron is connected to one or more neurons from the next layer. Each neuron from the last layer is connected to a number of output neurons [17] and holds a value. The value of the neuron gets tweaked during the training process. During the cutting state detection, each neuron's value performs as a coefficient of the neural network equation.

The values of each neuron could be determined by,

$$\forall\, \boldsymbol{j, l}\colon \boldsymbol{a_j^{(l)} = \sum_{i=1}^{n} w_{i,j}^{l} a_i^{l-1} + b_l b_j^l} \tag{39}$$

Where $a_i^{(l)}$ is the value of neuron i from layer l, $w_{i,j}^l$ is the weight which connects neuron i from layer $l-1$ and neuron j from layer l, b_i is the value of bias of layer $l-1$, and b_j^l is the bias weight of layer l and neuron j. Please note that in the above notation, layer 0 is the equivalent of the input. Using a feedforward ANN with 3 layers, 9216 inputs and 4096 neurons in each of its two hidden layers, its output is given by,

$$\boldsymbol{a^{(3)} = (w^3(w^2(w^1 x + b^1) + b^2) + b^3)} \tag{40}$$

Where w^l, b^l, and x_i, the input number i, are given as,

$$w^1 = \begin{bmatrix} w^1_{1,1} & \cdots & w^1_{1,9216} \\ w^1_{2,1} & \cdots & w^1_{2,9216} \\ \vdots & \ddots & \vdots \\ w^1_{4096,1} & \cdots & w^1_{4096,9216} \end{bmatrix}, \quad w^2 = \begin{bmatrix} w^2_{1,1} & \cdots & w^2_{1,4096} \\ w^2_{2,1} & \cdots & w^2_{2,4096} \\ \vdots & \ddots & \vdots \\ w^2_{4096,1} & \cdots & w^2_{4096,4096} \end{bmatrix},$$

$$w^3 = \begin{bmatrix} w^3_{1,1} & w^3_{1,2} & \cdots & w^3_{1,5} \\ w^3_{2,1} & w^3_{2,2} & \cdots & w^3_{2,5} \\ \vdots & \vdots & \ddots & \vdots \\ w^3_{4096,1} & w^3_{4096,2} & \cdots & w^3_{4096,5} \end{bmatrix}, b_l = \begin{bmatrix} b^l_1 \\ b^l_2 \\ \vdots \\ b^l_{4096} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{9216} \end{bmatrix} \tag{41}$$

During the training process, the feed-forward ANN's weights and biases are tweaked using backpropagation to minimize the network error. After training the network, it is able to identify the state with the highest probability from the output layer. For instance, in the case of the input signal be a chatter signal, output values $a_1^{(3)}, a_2^{(3)}, a_3^{(3)},$ and $a_4^{(3)}$ values (probability of air cut, tool entrance, tool exit) will be close to zero whereas $a_5^{(3)}$ will tend to unity, i.e. the probability of chatter being in process is high.

A regularization method called dropout, introduced in the ImageNet Classification with Deep Convolutional Neural Networks [30], is employed in the neural networks training implementation. Dropout sets t of each artificial neuron to zero with a given probability. This component increases the robustness of the learning by ensuring that different combination of the neurons is functioning. A drop out of 50% is implemented in this work. It means that in each iteration of training, dropout sets roughly half of the $a_j^{(l)}$ from equation (40) to zero. This results in a fitted equation during the training process in which each individual neuron ($a_j^{(l)}$) has a meaningful contribution to the machining state detection since it is fit to remain relatively functional in the absence of half of the other neurons.

Moreover, the output of each neural network layer goes through ReLU.

3.4.4 SoftMax

SoftMax turns the numbers from the output of the ANN to the probabilities, scaled from zero to one where one is 100% probability. For each machining state, the probability of that state for a given input is calculated by,

$$\boldsymbol{p}(\mathbf{state}) = \mathbf{SoftMax}\left(\boldsymbol{a}_{\boldsymbol{i}}^{(\boldsymbol{L}_{\mathbf{last}})}\right) = \frac{\boldsymbol{e}^{\boldsymbol{a}_{\boldsymbol{i}}^{(\boldsymbol{L}_{\mathbf{last}})}}}{\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{K}} \boldsymbol{e}^{\boldsymbol{a}_{\boldsymbol{j}}^{(\boldsymbol{L}_{\mathbf{last}})}}} \tag{42}$$

Where $a_i^{(L_{\text{last}})}$ is the i-th neuron of the $last$ layer and $K = 5$ is the number of machining states. $L_{\text{last}} = 3$ is the number of layers.

3.4.5 Focal Loss Classifier

The ANN output goes through a classifier to associate the output numbers to its corresponding classes', i.e. machining states, probability. In the machining processes, usually, the time which tool spend in-cut is longer than the time it spends getting into or out of the cut. The training dataset used to train follows the same pattern. It takes the tool a short time to pass the transient state of entering and exiting the workpiece. Hence, the number of entrance and exit instances is less than the number of other instances. It is crucial to compensate for this imbalance in the count of data from each state of machining to avoid having a trained cutting state prediction biased against detecting entrance and exit states or not being fitted well enough to detect the right characteristics which represent the mentioned transient states.

The focal loss function [32] multiplies the cross-entropy function with a modulating factor. It increases the sensitivity of the CNN to misclassified machining state instances.

In this thesis probability of each state of machining is denoted by p_{state}, where "state" could be replaced with any of the five cutting states.

The Focal loss (FL) for the probability of each training instance is given by,

$$\mathbf{FL} = -\boldsymbol{\alpha}(\mathbf{1} - \mathbf{p}_{\mathbf{state}})^{\gamma}\mathbf{log}(\boldsymbol{p}_{\mathbf{state}}) \tag{43}$$

Where $\alpha = 2$ is a constant called balancing parameter, $\gamma = 0.25$ is a constant called focusing parameter that specifies the system's sensitivity to wrong machining state detection, and p_{state} is the probability of the real state. The total focal loss for a set of instances is given by averaging the values of focal loss over all of the instances.

3.4.6 Transfer Learning (base) Method's Architecture Overview

As illustrated in Figure 26, the base network consists of five convolutional blocks, and each incorporates a number of layers [30]. The output of each block goes through a rectified linear unit.

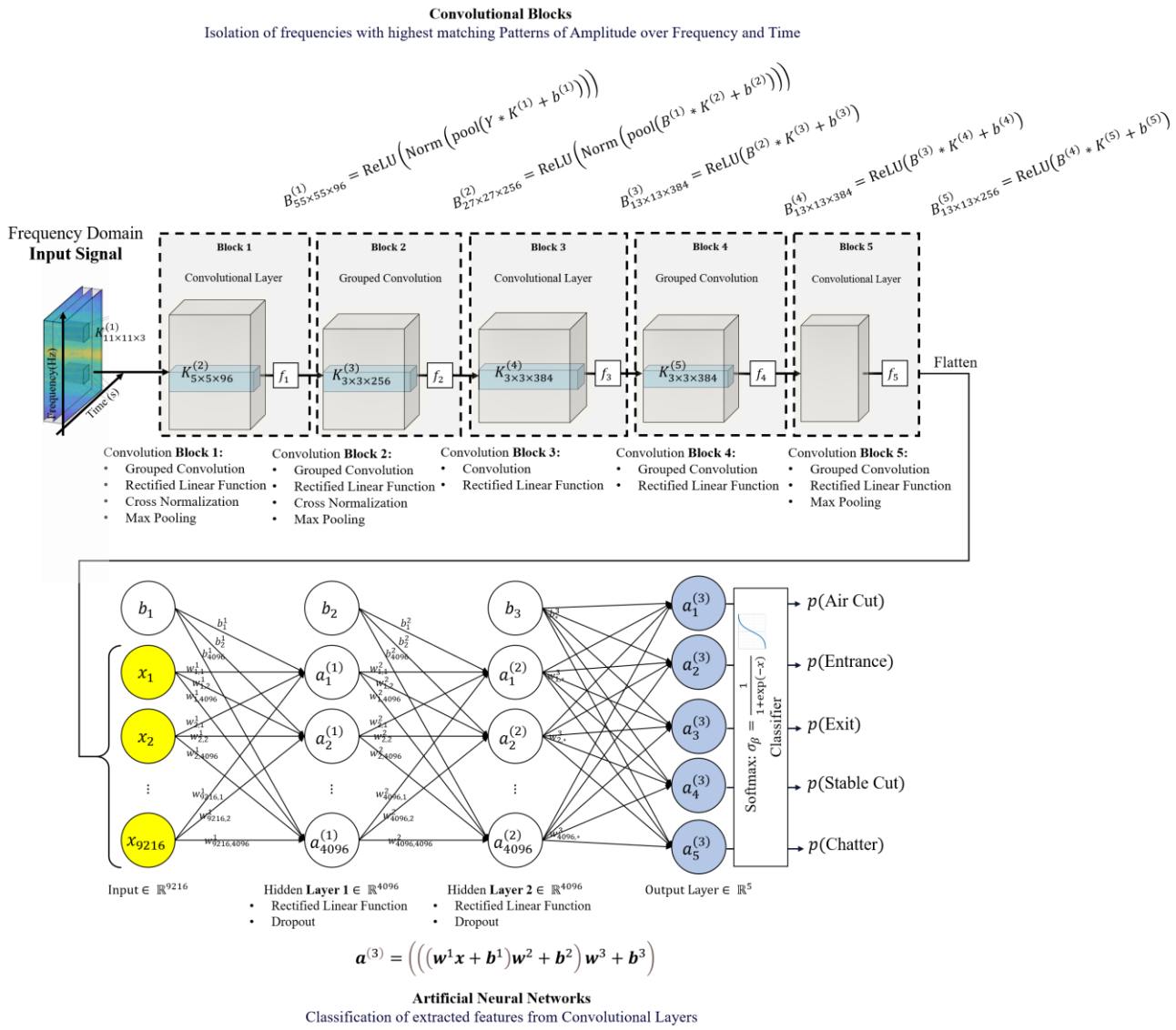

Figure 26 Transfer Learning Architecture Overview

Where f_i $(i = 1,2, \dots ,5)$ is the arithmetic and activation function of block i and Y is the input STFT instance. This section explains the arithmetical and logical function of each of the architecture layers, as well as the, clarifies and their benefits to this specific application. As shown in Figure 26, the input to the first block is the frequency domain signal during the time, generated by the method explained in section 3.3. The resultant matrix is resized to the network input size (227 by 227) using bicubic interpolation and triplicated to accommodate the transfer learning architecture's

input size. Bicubic interpolation determines each matrix element value by calculating the weighted average of the nearest four neighbours. In the case of a custom design network with an input size equal to the feature extraction step's output size, the last two mentioned steps are not required. Each block's input goes through a series of functions in that block, as denoted under each block and layer in Figure 26. The output of one block is the input to the next block. Each layer l, has a bias term of $b^{(l)}_{h_b^{(l)}}$ which has not been shown in Figure 26 to avoid clutter.

The output of the last convolutional layer is a two-dimensional matrix. The matrix gets flattened and goes to the ANN The output of the last convolutional block is classified into the machining states using three layers of ANN. The cross-entropy loss function from the original network architecture is replaced with a focal loss function due to an imbalance in the data. The focal loss function makes up for the fact that the number of the entrance and exit transient state instances is smaller than the other instances.

This section goes over the arithmetic building blocks of the convolutional blocks of the deep learning architecture used in this thesis. The part that each component of the network takes in the machining state detection and classification of the form attribute of the frequency-domain signal is explained. The form factor attributes of the input are the trend of changes in the frequency-domain signal amplitude in different frequencies over a period of time.

The convolution blocks shown in Figure 26 consists of the following layers. The first convolution block consists of an 11 by 11 convolutional layer of 96 convolution kernels with a stride of 4 and zero padding, a linear rectified function, a cross-channel normalization, and a max-pooling of size 3 by 3 and stride size of 2 and a zero padding. The input of the second Convolutional Block is the

output of the first block. It incorporates a grouped convolution layer of 2, 128 kernels with a filter size of 5 by 5 and a stride of one and a padding of 2, a rectified linear function, cross-channel normalization, and a max-pooling. The third Convolutional Block has a convolution layer of 384 and a filter size of 3 and a stride of one and a padding of one, and a rectified linear function. The fourth Convolutional Block has two groups of convolution layers of 192 and a filter size of 3 and a stride of one and a padding of one, and a rectified linear function. The last Convolutional Block has a convolution layer of 256 convolution kernels, a rectified linear function and a max pooling. During the training stage of transfer learning, the last convolution layer has been retrained.

The flattened output matrix of the convolutional layer goes to the ANN layers. The base model has a fully connected feedforward ANN. Its size input is 4096, equal to the output size of the last convolutional block's output. The first layer after the input has 4096 neurons as well. Input and the first layer are each followed by a rectified linear function as well as a dropout with a 50% probability. The output layer has five neurons, each corresponding to one of the states of cut. The outputs go through a Softmax function to give the probability of each cutting state. Input and the first ANN layer have dropouts, and they are followed by a rectified linear function. All the ANN layers have been retrained during the training process. The last layer of ANN is followed by a SoftMax layer. Hence, the overall architecture and the first three convolutional blocks remain as it is suggested in the original paper of Image Net. The last two convolutional layers and the neural network is retrained with the weights and biases of the image Net as their initial values. The loss function is changed to the focal loss to make up for the imbalance in the number of each machining state present in the training set.

The network is trained on the training data set using Adam [44] optimizer. Adam is a scholastic optimization method that uses the following moving averages to update the network parameters in i-th iteration, Θ_i, and calculate Θ_{i+1},

$$\boldsymbol{\Theta_{i+1} = \Theta_i - \frac{\alpha m_i}{\sqrt{v_i} + \epsilon}} \tag{44}$$

Where α is the learning rate, ϵ is the denominator offset, and m_i and v_i are parameter gradients and its squared, respectively, and are given by,

$$\begin{cases} \boldsymbol{m_i = \beta_1 m_{i-1} + (1 - \beta_1)\nabla E(\theta_i)} \\ \boldsymbol{v_i = \beta_2 v_{i-1} + (1 - \beta_2)[\nabla E(\theta_i)]^2} \end{cases} \tag{45}$$

The performance of the trained model is presented in section 5.3.

Further, during an automatic machine learning process [33], the base architecture evolves to make an architecture specialized in machining state detection.

3.5 Automated Convolutional Neural Network Architecture Design for Machining State Detection

The Architecture of Convolutional Neural Network plays a crucial role in the deep learning algorithm, performance, speed, and robustness. While transfer learning (introduced in section 3.4) offers a quick and robust network for cutting state detection, it does not essentially provide the best architecture in terms of being optimized for a specific purpose, in this case machining state detection. This section suggests a novel approach towards automated CNN design architecture methodology for designing a specialized CNN for machining state detection based on the frequency domain vibration signal. The work in this section results in a machine learning

architecture design for processing the vibration signals for the purpose of machining state detection with high accuracy and low model complexity.

Transfer learning comes with limitations and shortcomings. The idea behind the proposed approach is to have the ImageNet [30], also known as AlexNet, architecture, used for the transfer learning method, as the initial state and shape it according to the machining state detection problem. The base network (ImageNet) is modified based on the specification of cutting state detection. Afterwards, the customized network is used as the initial condition for the architecture design algorithm. Finally, using the proposed Automated Machine Learning (AutoML) technic a deep learning architecture is proposed. The proposed architecture is designed with three main goals in mind: high accuracy in machining state detection, low architecture complexity, and high explainability. The output of this section, including the designed network and the trained model's accuracy, is presented in section 5.4.

In terms of machining state detection accuracy and network performance, the automated architecture design uses Bayesian optimization to reach the mentioned set of goals. Moreover, a custom classification layer is designed to accommodate the requirements of real-world cutting.
Regarding complexity, the automated machine learning procedure is designed to have a high tendency to keep the complexity of the architecture low. The complexity of a deep learning architecture is defined as the number of its learnable parameters. The learnable parameters of a deep learning algorithm are the parameters that get adjusted during the training process. More training parameters indicate a more complicated equation to fit the cutting state detection problem in hand. The low complexity of the architecture is important for several reasons. First and foremost,

increasing the complexity of a network increases its inclination to overfit and misclassification of the machining states due to redundancy of the learnable parameters. Additionally, less learnable parameters result in a faster network, both during training and chatter detection, and a lighter model with regard to occupying space on the memory.

With respect to explainability, the input has not been resized since the designed CNN is specialized in the machining state detection need. On the contrary, in transfer learning, the input size of the network is given by ImageNet. Also, tooth passing frequency is used as one of the inputs to the network. Moreover, a custom loss function is designed to prevent overfitting and minimizing false positive chatter detection.

There are two significant concepts in the automated machine learning architecture design proposed by this thesis: Architecture Design Methodology, and Objective function and Network Assessment. Further, in this section, each concept is explained.

3.5.1 Automated CNN Architecture Design Methodology and Structure

A scheme of the automated architecture design is presented in Figure 27. The utilized architecture outline is similar to the transfer learning architecture from section 3.4, with minor changes to make it tailored for the real-world physical problem at hand.

A model is generated based upon a set of proposed architecture parameters during each iteration of the optimization algorithm. The model gets trained, and the performance of the trained model gets assessed. Based on the assessment, a set of parameters is generated for the next iteration of the optimization algorithm.

A methodology is proposed for automatically generating new architectures and modifying them to find a low complexity, high accuracy architecture. For each designed architecture, regardless of its specification, the input to the system will always be the frequency domain vibration signal as well as the tooth passing frequency, and the output will be the probability of each machining state. Based on the optimization algorithm's output, the best performing model gets trained on the training data and further used as the final machining state detection.

Below, a detailed overview of the automated architecture design is shown; each part of the proposed methodology is further elaborated in this section.

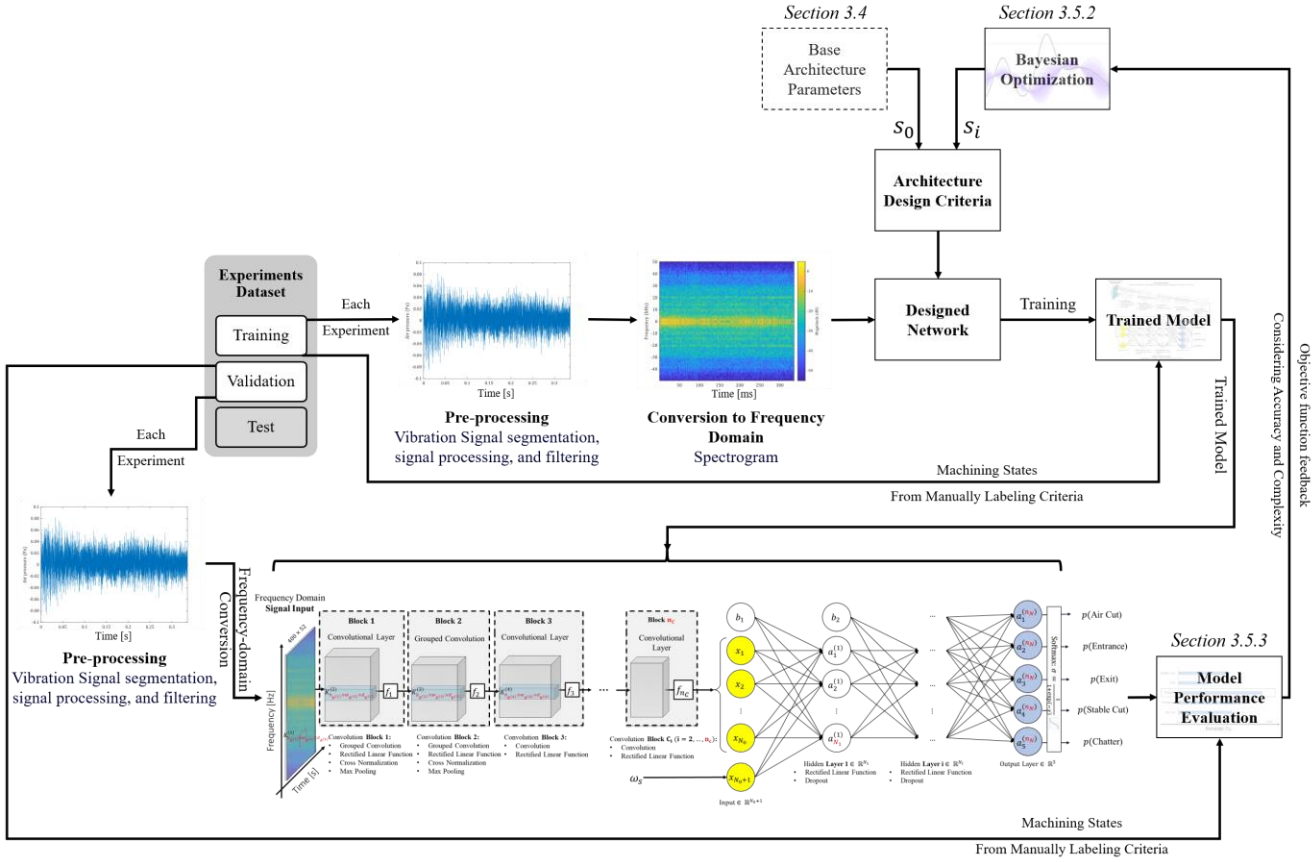

Figure 27 Overview of Automated Cutting State Detection Network Architecture Design Methodology Scheme

3.5.1.1 Architecture Structure and Outline

The initial architecture or the initial input (s_0) to the optimization algorithm is based on ImageNet from section 3.4. As shown in Figure 27, three slight modifications have been applied to ImageNet to make it suitable for cutting state detection.

First, the input size is customized to the frequency domain features input given in section 3.3 (400 by 52).

Second, the tooth passing frequency (ω_t) is added as one of the inputs to the neural network. Hence, the neural network inputs are the convolutional blocks' flattened outputs alongside the tooth passing frequency. Having tooth passing frequency as one of the network inputs leverages our knowledge in machining. Tooth passing frequency contributes to the formation of the frequency domain signal because of the resultant peaks in the tooth passing frequency and its harmonics. It makes it easier for the neural network to detect the cutting state. Since tooth passing frequency is a cutting feature, it increases the neural network state detection's robustness.

Finally, a customized weighted cross-entropy loss function with a penalty for false chatter detection is designed to accommodate the machining's particular needs. The loss function is given as below,

$$\mathbf{Loss} = -\sum \boldsymbol{W_P^T} \times \left(\boldsymbol{Y}.\mathbf{log}(\widehat{\boldsymbol{Y}})\right) \tag{46}$$

Where $Y$and $\hat{Y}$ are matrices of the actual state of the cutting instances, with the known state of machining, and the probability of each cutting state detected by the algorithm, respectively.

Furthermore, W_P is a matrix of miss classification penalty weight. It is designed based on the number of experiment instances for each machining state and the importance and effect of each type of misclassification. The elements of the mentioned matrices correspond to air cut, entrance, stable, chatter, exit cutting states, respectively. Each element of the mention matrices corresponds to a machining state with the mentioned order. Each matrix element's value is the penalty for misclassifying the corresponding row's machining state to the corresponding column state. Each row of the matrix is normalized in the time of utilization.

$$W_P = \begin{bmatrix} 0 & 16 & 10 & 10 & 10 \\ 10 & 0 & 10 & 10 & 14 \\ 25 & 25 & 0 & 1 & 25 \\ 25 & 25 & 1 & 0 & 25 \\ 10 & 16 & 10 & 10 & 0 \end{bmatrix} \quad (47)$$

Where the rows and columns correspond to Air cut, Chatter, Entrance, Exit, and Stable, respectively. As it is shown, the penalty for misdetection of two transient states of entrance and exit, which have a smaller number of training cases, is higher than the others. Additionally, all the penalties associated with false detection of chatter is 16 or higher. Same applies to mistaking stable for chatter.

Except for what was mentioned, the initial network which Bayesian optimization starts with the same topology and components as ImageNet.

3.5.1.2 Design Methodology

In the proposed architecture design methodology, each block has been considered as an individual entity with a set of hyperparameters. The number of the convolutional blocks and the neural network layers are subjected to be determined by the optimization algorithm. As shown in Figure 27, each convolution block is denoted by C_i $(i = 1, 2, \dots, n_c)$ where n_c is the number of the

convolution blocks, and the number of neurons in i-th hidden layer of neural network layer is denoted by N_i $(i = 1, 2, \dots, n_N)$ where n_N is the number of the neural networks hidden layers. Two $\mathrm{n_c}$ and n_N are determinant of the network topology.

In iteration i of Bayesian optimization, it comes up with a set of design hyperparameters, s_i which includes the network topology hyperparameters as well as the convolutional blocks and neural network hyperparameters. Each s_i is a unique identifier of a CNN architecture. The s_i set consists of $2 \leq n_C$ as the number of convolutional blocks and the size and the number of kernels of each of n_C convolutional blocks' number of kernels d_i and kernel size $(w_i \times h_i)$ for layer i as well as n_N as the number of hidden layers and the number of neurons in each of n_N layer (N_i). After having the s_i defined by the optimization algorithm, the corresponding CNN structure gets formed, and the feasibility of the CNN structure, in terms of matrix dimensions agreements gets assessed. In case of finding the proposed architecture not feasible, the Bayesian optimization acquisition function modifies its estimate using the method suggested by M. Gilbert et al. [34].

For each given set of n_c and n_N, the number of optimizable hyperparameters is a function of the number of layers by $2\mathrm{n_c} + n_N$, since each convolutional block has two parameters for width and height and each ANN layer has a number layers parameter. Additionally, n_c and n_N are subject to being optimized. Considering that the Bayesian optimization optimizes a set number of variables, a set of optimization variables for a maximum of five extra convolutional and neural layers are defined. In the optimization iterations with $n_C < 5$ or $n_N < 5$ the hyperparameters corresponding to layers after the n_c-th layer or n_N-th layer becomes irrelevant. The irrelevant variables were not taken into account during the optimization process, as it is suggested by K. Swersky [35].

The weights and biases associated with the first five convolutional blocks and the first two neural network hidden layers are initialized by resizing the weights and bias matrices of the trained transfer learning method, as suggested in section 3.4, using bicubic interpolation. This helps with faster convergence and more accuracy. For the rest of the weights and biases, the initialization strategy is the same as it is suggested for the last convolutional block and hidden layer of ImageNet [30]. The weights for the extra convolution layers are initialized from a zero-mean Gaussian distribution with a standard deviation of 0.01. The neuron biases of the additional convolutional layers and the hidden layers are set to constant one, and the neuron biases are set to constant zero. The number of epochs is limited to three during the optimization to minimize the architecture optimization time and saving computing power. Each epoch is a pass during which the whole training dataset is used to train the model.

Figure 28 shows the general rules according to which the architectures are designed during the optimization process. Other than the architecture's overall topology and a set of rules and constraints mentioned in this section, the rest of the architecture can be modified during the design methodology. The parts denoted with red font in Figure 28, are the tweakable part of the network, i.e. s_i contents in each iteration.

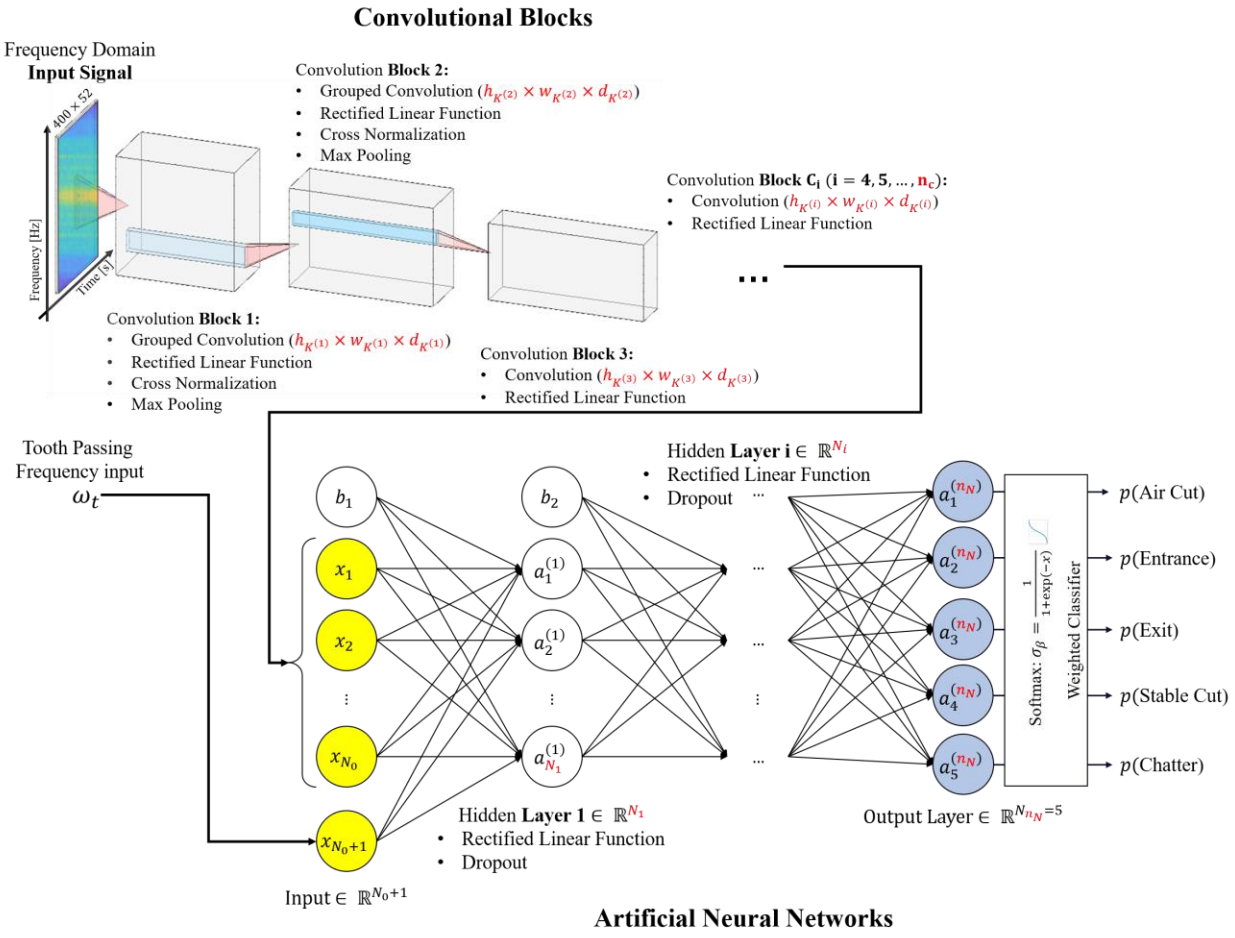

Figure 28 Architecture Structure and Design Methodology Scheme

3.5.2 Bayesian Optimization for Machining State Detection Architecture Design

Bayesian Optimization is a Global Function Optimization method that uses a response surface methodology-based approach in order to find a local minimum or maximum of an objective function. In this chapter, Bayesian optimization is incorporated as an approach to design a convolutional neural network specialized for machining state detection.

Bayesian Optimization is a proper method to be applied to the architecture design problems since they are particularly required to find the extrema of an objective function, which is costly, non-convex, and we do not have access to the derivatives [38].

Bayesian Optimization uses the Bayes theorem to form a surrogate probabilistic model of the objective function. The objective function is the function that is desired to be minimized or maximized; it encompasses the variables which contribute to the problem at hand. Since the use of optimization in this thesis is for the minimization of the objective function; hence, minimization will be the only case that will be explained in this thesis. The objective function definition is explained in the next subsection.

Surrogate optimization attempts to balance the trade-off of exploration and exploitation. Exploitation is sampling, according to the surrogate model's prediction. Exploration is the search for a global minimum where the surrogate model has an extremely low certainty, resulting in exposure to new local minima. The algorithm has been proven to converge [39].

Bayesian optimization is used to define a surrogate function that approximates the original function with computational efficiency. Bayesian optimization makes use of the prior estimation of the surrogate function. It updates based on the samples drawn from it in order to achieve a more accurate approximation of the surrogate model. For a function $f(x)$, the surrogate function is denoted by $\hat{f}$. Based on the fact that the defined objective function is complex, a non-parametric method would be appropriate to approximate the functions. In this work, Bayesian optimization constructs a prior belief about how the proposed architecture, associated with s_i set of network architecture hyperparameters, would behave. Then, in each iteration, it searches the parameter space by enforcing and updating that prior belief based on the current measurements. In other

words, a surrogate model of the problem search space, which has a size of about 10^{38} is formed using Bayesian optimization. The model gets evolved in each iteration of the optimization to make a more accurate picture of the search space. The search for the best architecture will be carried on based upon the updated surrogate function, from the previous iteration.

For the purpose of sampling the points in the search space, Bayesian optimization uses a function called acquisition function, which is denoted by $\mu_Q(x)$. An acquisition function maintains the balance between exploitation and exploration. Exploitation is searching the region in which the surrogate model predicts that the minimum values lay there. On the other hand, exploration is sampling at locations with high uncertainty.

The Bayesian optimization procedure starts with an initial definition of the prior for the surrogate model. Afterwards, in each iteration i, Bayesian optimization takes three steps towards optimization of a given objective function f:

1) Finding the sampling point x_i, based on the attributes of the posterior distribution and by using the acquisition function,

$$\boldsymbol{x_i = \mathbf{argmax}_x u(x|D_{1:i-1})} \tag{48}$$

Where u is the acquisition function, and Ds are the samples drawn from f.

2) Obtain a possibly noisy sample, y_i, from f as below,

$$\boldsymbol{y_i = f(x_i) + \epsilon_i} \tag{49}$$

3) Augmenting the data as below and updating the Gaussian process,

$$\boldsymbol{D_{1:i} = \{D_{1:i-1}, (x_i, y_i)\}} \tag{50}$$

The mentioned acquisition function evaluates the goodness of fit to a specific Gaussian kernel. The acquisition function used in this thesis is the probability of improvement. The kernel used in

this work is the ARD Matérn 5/2 kernel [40]. The estimate of the goodness will be modified based on the approach proposed by M.A.Gelbart et al. [34].

3.5.3 Network Assessment and Objective Function Design

The objective function evaluates two main criteria: network complexity and its ability to detect machining states. The objective function goal is to decrease the network complexity without compromising cutting state detection performance.

The complexity of the network is defined as the number of learnable parameters of the network. Each block of the deep learning system has a set of weights and biases that gets tweaked during the training process. The training phase's final weights and biases will form an equation in the test/implementation phase, which output each cutting state probability. It is desirable to have a smaller number of learnable parameters to make the equation as simple as possible. A smaller equation is faster to run, i.e. results in a faster chatter detection. Moreover, a simpler equation is less prone to overfit. The last section's resultant trained network performs cutting state detection on the validation data to evaluate the network performance. The mean of the network’s loss over all the predictions is calculated and used as an indicator of the design network performance. Lower loss indicates a better performance, i.e. smaller error.

Given the high volume of training samples and the fact that the objective function must account for network performance and its complexity, the Hannan–Quinn information criterion(HQC) [36] is used. The Hannan–Quinn information criterion (HQC) is given by,

$$\mathbf{HQC} = \boldsymbol{T}\mathbf{log}\left(\frac{\sum \mathbf{Loss}}{\boldsymbol{T}}\right) + \mathbf{2}\boldsymbol{k}_{\mathbf{complexity}}\mathbf{log}\big(\mathbf{log}(\boldsymbol{T})\big) \quad (51)$$

Where $\sum \text{Loss}$ is the summation of the loss function value for all of the training instances, an indicator of network performance, $k_{\text{complexity}}$ is the number of the network parameters, i.e. network complexity, and T is the number of training instances. It is common to use HQC in model selection and AutoML. Given the high number of training instances in deep learning, HQC is a proper candidate to be used [37]. Hannan–Quinn information criterion assesses the accuracy and complexity of the network by introducing a numeric measure. In other words, it measures the accuracy of the network, by evaluating the trained network on the validation data set, and adds a penalty for more complex networks.

3.6 Summary

In this chapter, the experiment design and machining state detection methodology are explained. The process of network design starts with using a statistical method for machining state detection to assess the plausibility of solving the problem of machining state detection using machine learning. A transfer learning method is used as the initial input set of the customized architecture design loop. An automated machine learning architecture design pipeline is proposed to design a specialized architecture for machining state detection with high accuracy and low complexity. It consists of processing vibration data and conversion to the frequency domain, as well as a machine learning model and constructor and Optimization components to detect machining states. During the architecture design process, the optimization algorithm suggests a set of architecture parameter of s_i and a network gets constructed based on them. The network gets trained on the processed signal with the training dataset. The trained model is evaluated using validation data. The objective

function value is assessed at this point, based on the accuracy of the trained model state detection on the validation dataset and the network's complexity. The best performing architecture is trained and used for machining state detection.

Chapter 4: Machine Learning and Physics-driven Chatter Detection Fusion

This chapter is dedicated to amalgamating the deep learning method for machining state detection from Chapter 3 and a chatter detection method based on milling physics and proposing the full system's pipeline. The goal is artificial intelligence and physics-driven methods infusion to reach a reliable approach with high accuracy.

The physics-driven method is based on an online energy-based chatter detection. Improvements suggested to increase its accuracy is explained in this chapter.

This chapter is organized as follows. The first section elaborates on the enhancements proposed to the energy-based chatter detection method proposed by Caliskan et al. [41]. The proposed enhancement increases the method's accuracy while it saves computing power required for detecting chatter. Section 2 focuses on the chatter detection pipeline and its implementation in the real world. A decision-making algorithm is proposed, which plays a primary role in the infusion process. Both deep learning and physics-based methods run simultaneously in parallel, detecting the machining state and stability independently. However, the decision-making module infuses the outcome of the mention methods. It results in a method with higher robustness and accuracy than the standalone machine learning method.

Section 3 suggests a self-evaluation loop for the machine learning algorithm, which uses the whole system's entropy and accuracy to retrain the model online and improve it. Lastly, section 4 provided a summary of the work done in the chapter.

4.1 Improved Energy-Based Chatter Detection Method

In this section, modifications are suggested to the online energy-based chatter by Hakan et al. [11]. The modification enhances the method by increasing the accuracy of the energy-based chatter detection method and speeds up the detection process. The original paper [11] provides an estimation of chatter and periodic frequencies based on the assumption of Kalman filter ideal performance. Then, it uses the estimated energy of chatter and periodic, forced vibrations, to detect instability in the cutting process.

This thesis proposes an approach to improve the robustness of the energy-based method. The proposed method estimates the chatter energy based on the chatter peaks detected in between tooth passing harmonics using a statistical approach. The mentioned method calculates/updates the FFT of the Kalman filter output. It avoids multiple filters and estimates the chatter energy more accurately.

As shown in Figure 29, subtracting the Kalman output from the input signal removes the input vibration signal's periodic part. In the proposed method, Kalman’s output is converted to the frequency domain continuously. The frequency-domain of the non-periodic parts of the vibration signal’s peaks are detected using a statistical method. The proposed peak detection method is designed to ignore the possible remaining peaks at tooth passing harmonics and detect the chatter peaks between the tooth passing harmonics' pairs. Finally, the energy ratio is calculated and used to detect chatter vibration. It is unstable if the energy ratio is higher than a predetermined threshold. Additionally, the parameters of the Kalman filter are tuned using Bayesian optimization.

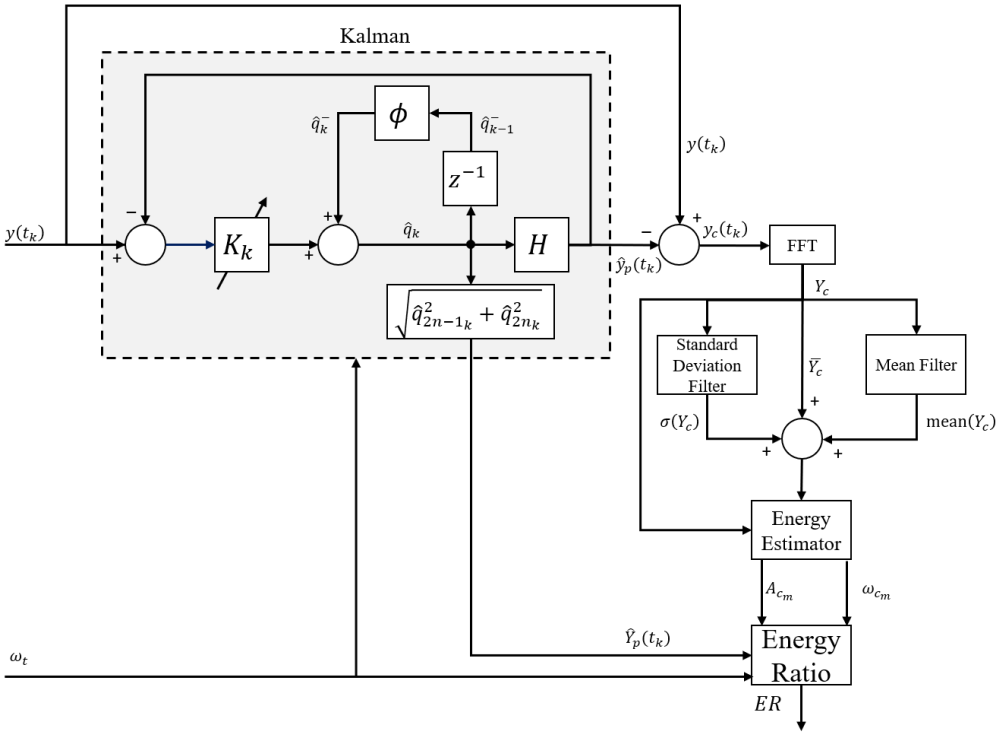

Figure 29 Suggested Improved Energy-based Method's Scheme

4.1.1 Chatter Energy Estimation

The input is the vibration signal, i.e. microphone in the thesis, which is passed through Kalman filter to predict $\hat{q}_k$ (Figure 29) . The state vector of the periodic part of the signal [12] is,

$$\begin{cases} \hat{\boldsymbol{q}}_{\boldsymbol{k}}^{-} = \boldsymbol{\phi}\hat{\boldsymbol{q}}_{\boldsymbol{k-1}} \\ \boldsymbol{P}_{\boldsymbol{k}}^{-} = \boldsymbol{\phi}\boldsymbol{P}_{\boldsymbol{k-1}}\boldsymbol{\phi}^{\boldsymbol{T}} + \boldsymbol{\lambda}\boldsymbol{R} \\ \boldsymbol{K}_{\boldsymbol{k}} = \boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{P}_{\boldsymbol{k}}^{-}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-\boldsymbol{1}} \\ \hat{\boldsymbol{q}}_{\boldsymbol{k}} = \hat{\boldsymbol{q}}_{\boldsymbol{k}}^{-} + \boldsymbol{K}_{\boldsymbol{k}}(\boldsymbol{s}_{\boldsymbol{k}} - \boldsymbol{H}\hat{\boldsymbol{q}}_{\boldsymbol{k}}^{-}) \\ \boldsymbol{P}_{\boldsymbol{k}} = (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{k}}\boldsymbol{H})\boldsymbol{P}_{\boldsymbol{k}}^{-} \end{cases} \quad (52)$$

where P_k is the error covariance matrix, K_k is the recursively updated Kalman gain matrix, R is the measurement noise covariance, Q is the diagonal process matrix of the process noise, and ϕ is the phase, as suggested in the energy-based method [10]. The energy of the periodic part of the signal ($\hat{Y}_p$) is calculated using Equation (19).

The Kalman filter's output is subtracted from the original signal to remove the effects of the periodic part of the signal at tooth passing frequency and its harmonics. The output of the Kalman filter is transformed into the frequency domain.

Ideally, the resultant frequency domain signal (Y_c) should not contain any periodic part of the signal. However, it is observed in the experiments that in a considerable number of cases, the harmonics of the tooth passing frequency are not totally removed. Since the Kalman filter predictions are based on the current state, changes in the cutting state or sudden changes in the cutting conditions affect the Kalman filter's prediction accuracy. The transient states of the tool entering and exiting the workpiece negatively influence the Kalman filter prediction. Hence, the following approach is proposed to increase the robustness of energy calculation.

As shown in Figure 30, for the purpose of chatter detection, a robust chatter peak detection in FFT of the vibration signal is suggested. The proposed method set a threshold curve on the FFT magnitude based on the local mean and standard deviation of the frequency-domain signal as well as the average of each instance of the frequency-domain signal's amplitude. The $n \times 1$ amplitude threshold value matrix for each of $n = 5000$ points of the frequency-domain signal is given by,

$$a_{th} = \sigma(Y_c) + \text{mean}(Y_c) + \overline{Y_c} \times J_{n,1} \quad (53)$$

Where $\sigma()$ and $\text{mean}()$ represent standard deviation and mean filter, respectively, both with a window size of $L = 0.004 \times n$. Moreover, $\overline{Y_c}$ is the average of the whole frequency-domain signal, and $J_{n,1}$ is a unit matrix by the size of the number of FFT amplitude points. Given the sudden change in the mean and standard deviation, a suppression method is suggested. The suppression factor for each amplitude point of Y_{c_i} with the threshold of a_{th_i} is defined as below,

$$f_{sup} = e^{\frac{\max\left(0, Y_{c_i} - a_{th_i}\right)}{\max(Y_c)}} \tag{54}$$

The suggested suppression factor decreases the effect of the peaks exponentially to increase the robustness of the detection. The suppression factor is exponentially proportional to the difference between each amplitude and the corresponding threshold. After applying the suppression factor, by the element-wise division of $Y_c./f_{sup}$, the threshold is recalculated using equation (53). For each group of FFT points with an amplitude higher than the threshold curve, i.e. each set of continuous points above the threshold, the point with the highest amplitude is determined as a candidate peak. The mentioned threshold is shown in Figure 30.

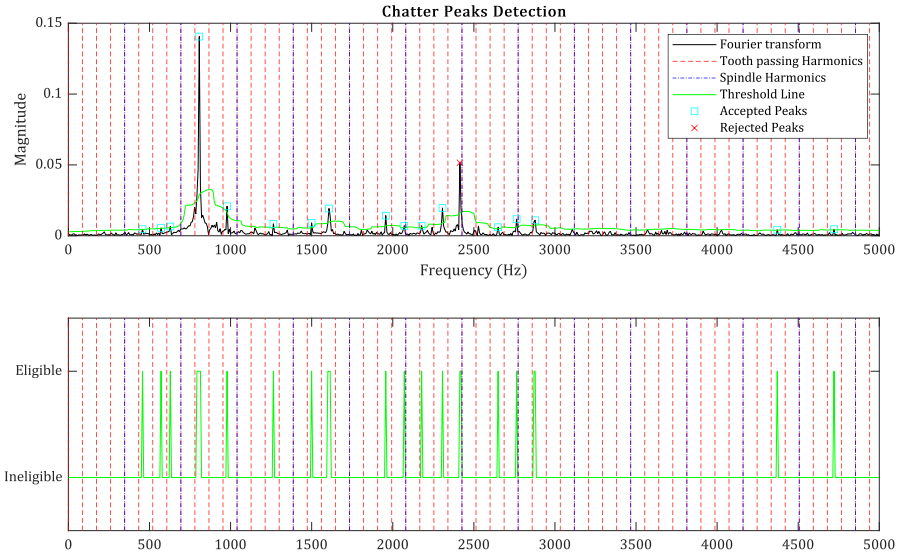

Figure 30 Chatter Peak Detection Example

An eligibility criterion is defined for each candidate to avoid the unremoved tooth passing peaks being counted towards chatter peaks. For each set of points higher than the threshold, its corresponding candidate peak is eligible if the set of the points does not cross any tooth passing frequency. As illustrated in Figure 30, the eligible points might appear in multiple ranges of

frequencies. The ranges that go through tooth pass frequencies are ignored. In each remaining range, the value points with the highest values determined as a chatter peak.

The proposed model's performance on three states of machining, chatter, entrance, and stable cut, for a sample signal is shown below. The magnitude of the detected peaks for chatter is considerably higher than the two other states. The detected peaks for the transient and stable states are mostly rejected.

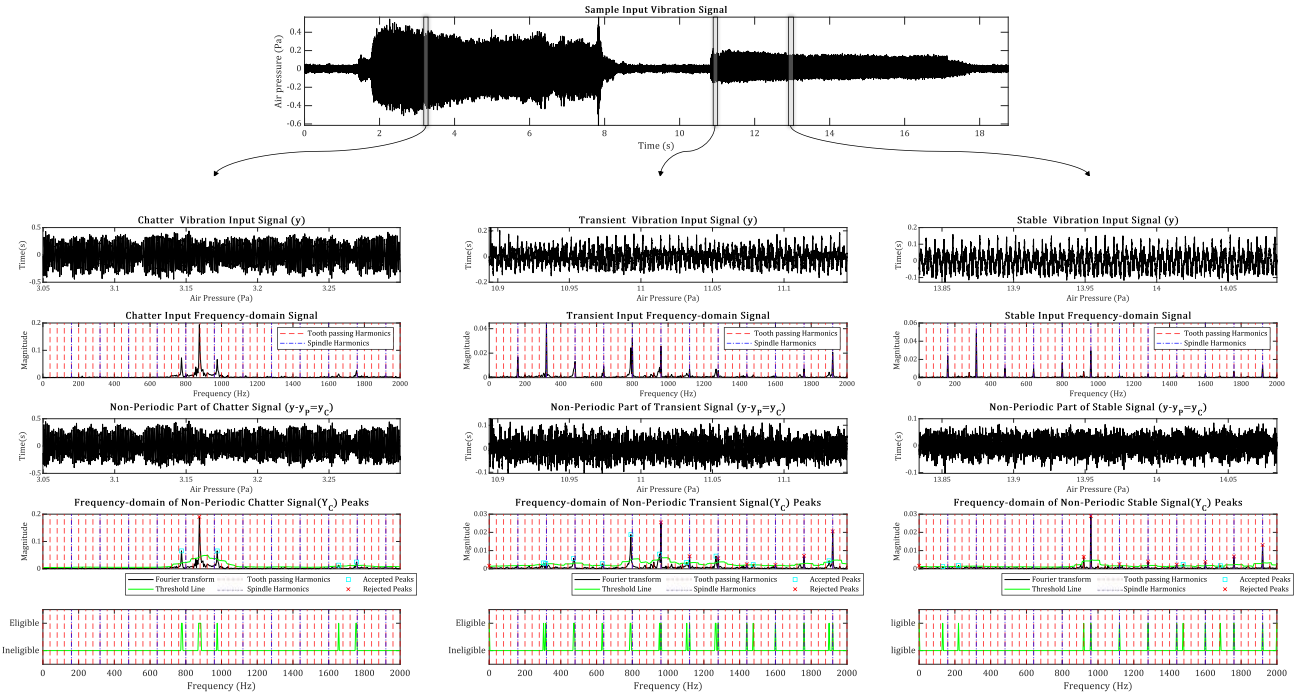

Figure 31 Proposed Physics-driven Method Performance on a Sample Signal

4.2 Hybrid Model

A decision-making process is designed for making the final call on chatter detection. The decision-making algorithm is the infrastructure of fusing the deep learning method output with the optimized physics-based method in order to detect chatter. If chatter is detected, the chatter

suppression module will get engaged and eradicate the cut's ongoing instability. The model is designed in a manner that it outputs the cutting state as well as the cutting stability. As shown in Figure 3 System Overview, the outputs of both physics-driven and machine learning methods are the inputs of the decision-making algorithm. Cutting state and a command to the machine tool controller, intended to perform chatter suppression, are the algorithm's outputs. Decision-making algorithm shares its inputs with the Self-evolution algorithm.

4.2.1 Decision-making Algorithm

The machine learning model output is in the form of the probability of each cutting state. The standalone deep learning algorithm determines the cutting state with the highest probability as the current cutting state.

As illustrated in Figure 32, the detected state's certainty is defined as the difference between the state with the highest probability and the one with the second-highest probability. The detection is with low certainty if the difference between the probability of the two states with the highest probability is smaller than the certainty threshold ($th_{\text{certainty}} = 8\%$). Intuitively, when the algorithm determines two relatively close probabilities for the top two most probable machining states, it means that the deep learning algorithm is agnostic about the correct state. In other words, low certainty shows the algorithm’s inability to differentiate between two cutting states.

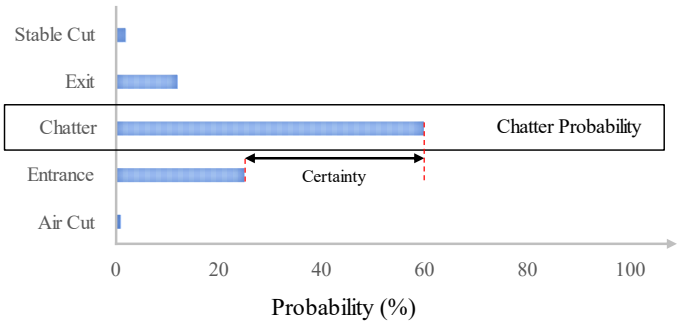

Figure 32 Probability versus Certainty in Machining State Detection Using Machine Learning Output

Energy-based chatter detection outputs the energy ratio of the signal. Based on the experimental results, an energy-based chatter detection algorithm has a high tendency to confuse the transient states of machining as chatter.

The outline of the decision-making algorithm is shown in Figure 33. If the deep learning algorithm detects Air Cut, Entrance, or Exit with a high certainty relative to a non-transient in-cut state, it is the final output of the decision making algorithm.

If the algorithm detects chatter or stable cut, the probability of the chatter and stable cut is given by,

$$\begin{cases} \boldsymbol{P_c = \dfrac{p(\text{Chatter}) + K_p \times EN}{K_p + 1}} \\ \boldsymbol{P_s = \dfrac{max(p(state \neq chatter)) + K_p \times (1 - EN)}{K_p + 1}} \end{cases} \tag{55}$$

Where P_s, P_C are the accumulative probability output of the deep learning algorithm, $K_p = 0.3$ is the energy-based influence factor, and EN is the energy ratio from energy-based chatter detection. If $P_C > P_S$ chatter is detected, and the chatter suppression procedure is initiated. Otherwise, the machining state will be determined as the state with the highest probability, given that it has at least $4 \times th_{\text{certainty}}$ of certainty. If it does not, the state will be determined stable.

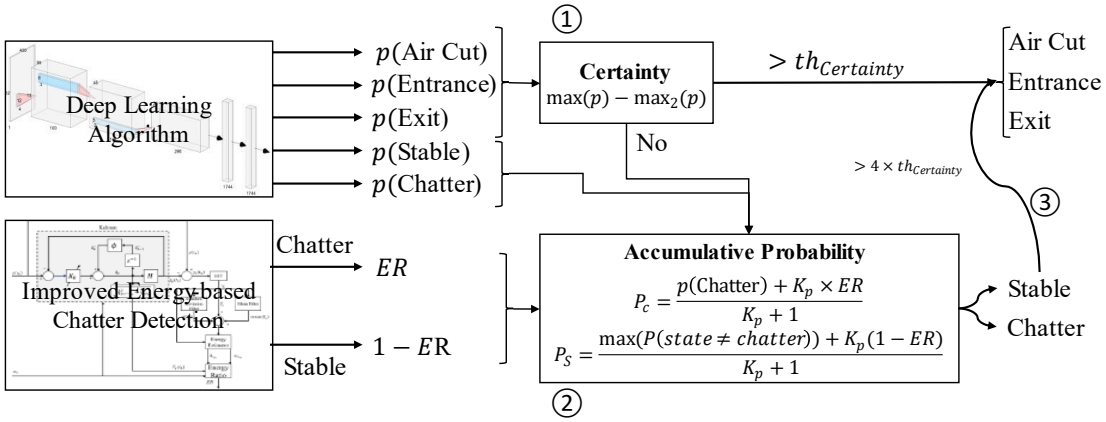

Figure 33 Decision Making Algorithm Outline

4.2.2 Chatter Suppression

After chatter is detected, the implemented method to suppress chatter will be engaged. The chatter suppression module's output is a new spindle speed, which is sent to the machine tool controller in case of chatter being detected.

In case of chatter being detected, a command is sent to the machine to zero the feed rate and change the spindle speed to the closest stable spindle speed, according to the stability lobes. As shown in Figure 34, the Sandvik tool chatters at 2100 [rev/min] and 2.8 mm depth of cut, despite what the theoretical Analytical Stability lobes suggest. Hence, the algorithm sends the command to the machine to zero the feed rate, set the spindle speed to 1665 [rev/min], and reactivate the feed rate.

This method requires the frequency response function of both the machine tool and the workpiece, cutting coefficients of the work material, and tool geometry to determine the stability lobes [7] beforehand.

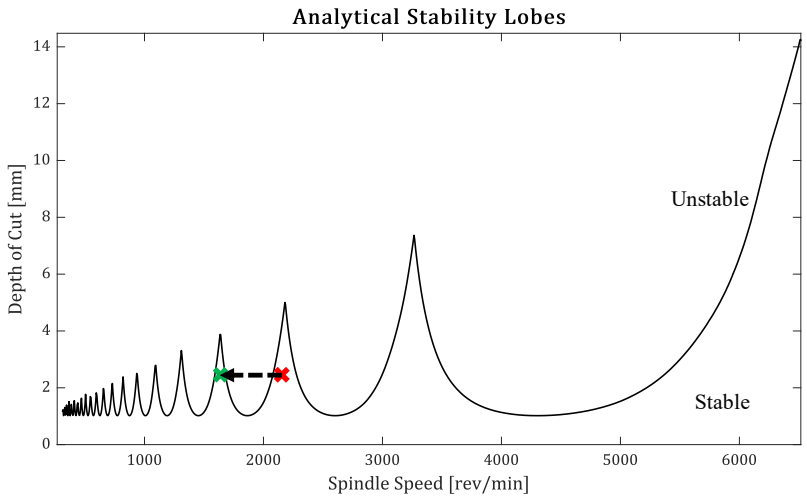

Figure 34 Chatter Suppression based on the Analytical Stability Lobes for Sandvik R390-050022 Tool

4.3 Self-evolving artificial intelligence using Semi-supervised learning

This section aims to develop a self-evolution mechanism to make the machine learning model to improve over time by self-training. There are three criteria proposed for automated retraining of the machine learning model. The first criterion is designed based on the machine learning model's attributes. The second one is designed based on machining process knowledge.

It is essential to mention the proposed method's goal is to find out an objectively accurate detection of the machining state for the instances detected with low certainty. The priority is to find and utilize more accurate detection at the cost of compromising the number of new training cases.

4.3.1 Self-evolution System

A self-evolving artificial intelligence system is capable of improving itself during operation. A semi-supervised learning is used for fabricating a self-evolving deep learning algorithm. The algorithm uses the fusion of the physics-based method and machine learning to retrain and enhance the convolutional neural network in an online manner. It takes advantage of the designed hybrid system to improve the machine learning model.

During the machining process, the network gets evolved while it is being used to detect the machining state. The system is capable of improving itself both on a pre-set criteria and with an operator's feedback.

N. Fazakis et al.[42] showed that using Semi-supervised learning improves machine learning algorithm performance merely by retraining the model on its predictions. Since the combination of the physics-based and deep learning algorithms have higher accumulative robustness and accuracy, the cutting-state instances detected with a high accumulative probability and accuracy using both methods are reliable sources of new training cases to retrain the machine learning model.

The goal is to retrain the machine learning model on the vibration signal instances that the machine learning model either misclassifies or correctly classifies with low certainty.

According to the defined criterion, each machining vibration measurement instance should meet at least one of the following three conditions to be qualified to be used for retraining and evolving the machine learning model.

First, if both physics-driven and machine learning methods have the matching stability detection, i.e. both detect chatter or physics-driven model detects stable cut while the deep learning model is detecting anything but chatter the certainty will be used to measure the reliability of the detection. Regardless of the certainty of the machine learning model's detection, if the energy ratio (EN) is higher than the defined th_{ER_C} threshold, the detection is chatter, data alongside the detected state will be used to retrain the model. If the detected state is stable and the EN is smaller than th_{ER_S}, the data alongside stable detection will be used to retrain the model. The th_{ER_C} and th_{ER_S} are the

energy ratio threshold for chatter and stable detection, respectively. The mentioned thresholds are determined by averaging the to 10% of chatter and the bottom 10% stable cases by the improved energy-based method.

This criterion becomes highly important and influencing when the certainty of the machine learning method's detection is low, i.e. the ML system is highly agnostic about its correct detection of machining state.

Secondly, each part of a tool path plan starts with air cut followed by an entrance of the tool to the workpiece. It continues to cut through the workpiece, either as stable or in the presence of chatter, and ends with the tool exiting from the workpiece. Knowing this typical characteristic of machining operations, if both systems detect chatter in one instance while it is surrounded by high certainty detection of chatter by the machine learning algorithm, higher than $th_{\text{Certainty}}$, as well as non- contradictory detection, the data alongside the chatter label will be used for retraining the model. In this context, surrounded is defined as having a hundred sampling before and after of the instance, meeting a particular criterion. Moreover, non-contradictory detection is when there is no disagreement between the physics-based method and the machine learning method detections.

Moreover, suppose the physics-driven method detects stable cut while the deep learning algorithm detects one of the machining states, other than chatter, either surrounded by the same state or the states that match the order mentioned in this paragraph. In that case, it will be used to retrain the model.

Finally, the machine's operator's feedback will be used as a way to retrain the algorithm.

4.4 Summary

In this chapter, proposes improvements to the energy-bases chatter detection and suggests an approach towards chatter detection by amalgamating the the physic-based and machine learning method. The proposed method is a fusion of the machine learning machining state detection model, designed and trained in the previous chapter, and a physics-driven chatter detection model based on H. Caliskan et al. [11] work with fundamental enhancements proposed at the beginning of this chapter.

The chapter starts by elaborating on the physics-driven chatter detection method suggested in this work and proceeds with the approach taken towards the infusion of the mentioned method with machine learning. The infusion method is proposed with robustness and accuracy in mind. Finally, a set of criteria is suggested to be used for fine-tuning the machine learning model as it is being used for state detection.

Chapter 5: Experimental Validation and Results

In this chapter, the experimental outcomes of each method, elaborated in chapters 3 and 4, are presented, and the functionality of the suggested model is verified based on experimental data.

The machining state detection machine learning models, improved energy-based chatter detection model, and the hybrid-system are validated with experimental data. The experiments used for reporting the model's performance is from a new set of data associated with the measurements and experiments that have not been used during the model design or training. Formerly, in this thesis, referred to them as the test data set. It consists of cutting tests with a wide variety of cutting conditions, including cutting conditions different from what is used for model training and design. The data consists of cutting experiments that follow the theoretical linear stability laws presented by Y.Altintas [6] and those that contradict them, i.e. chatter in the theoretical stable zone and vice versa.

Each section of this chapter represents a fully-working original model for chatter detection. The chapter is organized as follows. It starts with an overview of the performed experiments. In the second section, the set baseline and its accuracy are shown. Section 5.3 explores the capability of the model resulted from the transfer learning method introduced in section 3.4. Section 5.4 evaluates the performance of the designed machine learning model in section 3.5. Section 5.5 presents the enhancements caused by the improvements applied to the energy-based method, as suggested in 4.1, and validates the final hybrid model's performance, introduced in section 4.2. Finally, section 5.7 summarizes the outcomes of this chapter.

5.1 Data and Training Parameters

The first step towards having a robust model and avoiding overfitting is to have a diverse dataset. For this purpose, more than 1600 experiments from about 500 continuous cutting tests, consist of various cutting conditions, tool paths, tools, and materials, are performed. The data is acquired with $f_s = 50kHz$ sampling frequency and resulted in 53,884 training instances. The cutting instances were extracted and labelled to five machining states manually based on the process output and surface quality.

Table 1 Number of Instances and Experiments Distribution for each Dataset and each Machining State

	Training Instances	Validation Instances	Test Instances	Total (%)		Experiments (%)	
Air cut	14,110	1,200	4,048	19,358	(35.90%)	464	(28.10%)
Entrance	540	112	130	782	(1.50%)	286	(17.32%)
Stable	16,364	1,280	1,272	18,916	(35.10%)	358	(21.68%)
Chatter	9,806	1,966	2,292	14,064	(26.10%)	218	(13.20%)
Exit	480	182	102	764	(1.40%)	325	(19.69%)
Total (%)	41,300 (76.60%)	4,740 (8.80%)	7,844 (14.60%)	53,884			
Experiments (%)	1256 (76.08%)	190 (11.51%)	205 (12.42%)				

The cutting experiments differentiated in parameters such as workpiece material, tool, and cutting conditions. The workpieces' materials used are aluminum 7075 and steel. In terms of tools, as shown in Figure 35, Six different tools have been used during the data collection as follows, Sandvik R245-12T3, Seco R21769-1632, Sandvik R390-050022, Seco JS452120E33R050, and Sandvik 39241014-6340120B. The Seco R21769-1632 tool is used for cutting steel, and the rest of the tools are tools used to cut aluminum workpieces.

The data collected from chattering and stable cuts have been conducted at various depths of cuts, feed rate, and spindle speed. All of the tools were tap tested to identify their Frequency Response

Functions (FRF) while they were mounted on a warmed up machine. The tap test data further used to determine the analytical stability lobes [6] and to plan the experiments accordingly. For each tool, several pairs of depth of cut and spindle speed are measured from each of the four following categories, unstable cut in theoretically stable zone, unstable cut in theoretically unstable zone, stable cut in theoretically unstable zone, and stable cut in theoretically stable zone. The tests cover the predicted stability, unpredicted stability, predicted chatter, and unpredicted chatter conditions. The mentioned statement applies to all of the tools, except Sandvik R245-12T3 and Seco R21769-1632.

Figure 35 The tools used during data collection, from left to right Sandvik R245-12T3, Seco R21769-1632, Sandvik R390-050022, Seco JS452120E33R050, and Sandvik 39241014-6340120B

5.2 Baseline

The baseline is set in section 3.2.2 to assess the machining state detection problem's plausibility using statistical methods. The output model is a machine learning model, capable of machining state detection independently.

For the goal of machining state detection, a binary decision tree is utilized. The tree uses Gini's diversity index as its split criterion [43]. As mentioned before, cross-validation with ten folds is used on 0.2 seconds pieces of training data.

The method resulted in 85.46% test accuracy in cutting state detection. The Confusion Matrix of the trained model is shown in Figure 36, demonstrating the misclassifications of each cutting state. The Green and red colour scale in the confusion matrix indicate the correspondence class prediction accuracy, where green is an indicator of high accuracy, and red is an indicator of low accuracy. To make the matrix more readable, each row of the matrix represents the percentage regarding the true class of the corresponding row.

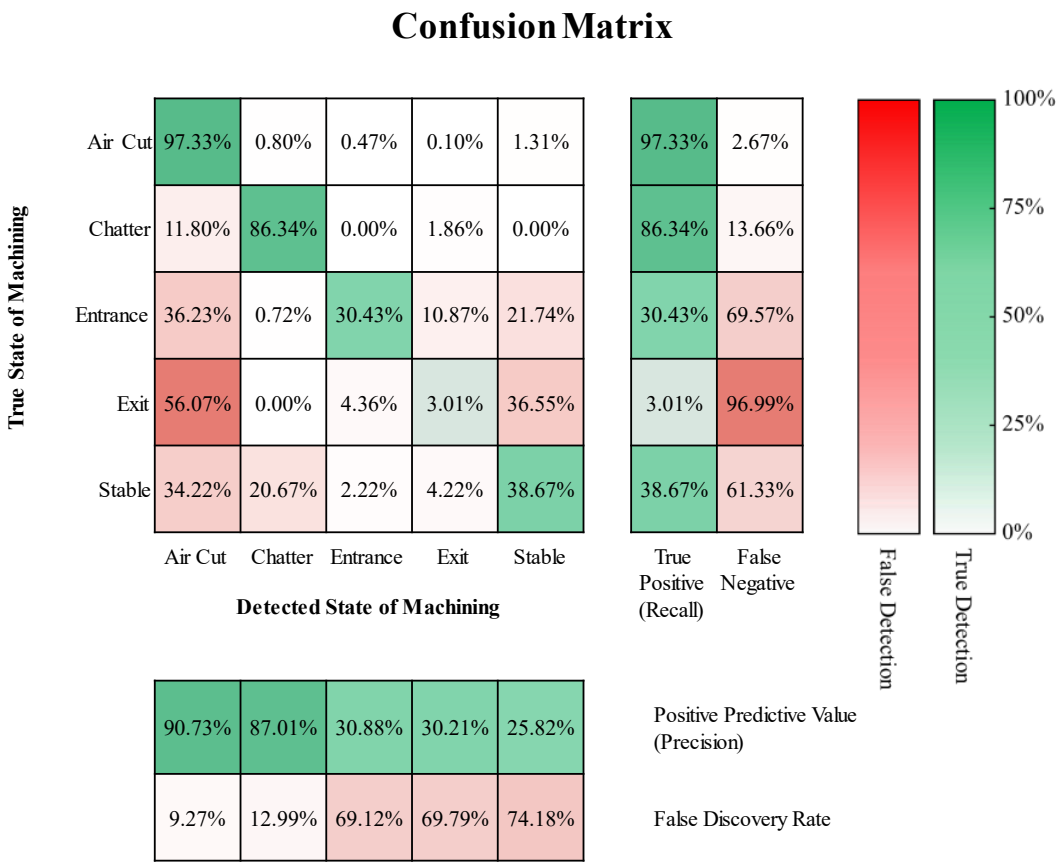

Figure 36 Baseline Model Confusion Matrix (with normalized rows)

The model shows promising results in terms of chatter detection. The model is not well trained for detecting entrance and exit of the tool in the workpiece, yet, it rarely mistakes entrance and exit states with chatter. In other words, the model shows substantial results in chatter detection as well as machining state detection. The algorithm performance in detecting entrance and exit is relatively low, however it has about 10% higher accuracy than a random detector in detecting entrance..

In Figure 37, the trained decision tree is depicted, and the range of statistical variables indicating each machining state is shown whereσ indicates standard deviation, and $\tilde{\mu}_3$ stands for Skewness. For each split, the left arm represents the case in which the variable in the corresponding box on the top of the branch has a smaller value than the number indicated in the box. Similarly, the right arm leads to where the variable, i.e. Standard deviation or Skewness as specified in the box, is bigger or equal to the stated number in the same box. Starting from the top of the tree, the value of the standard deviation and skewness of the vibration signal is compared to the value in the corresponding box and navigates its way to one of the end nodes, which are the machining states.

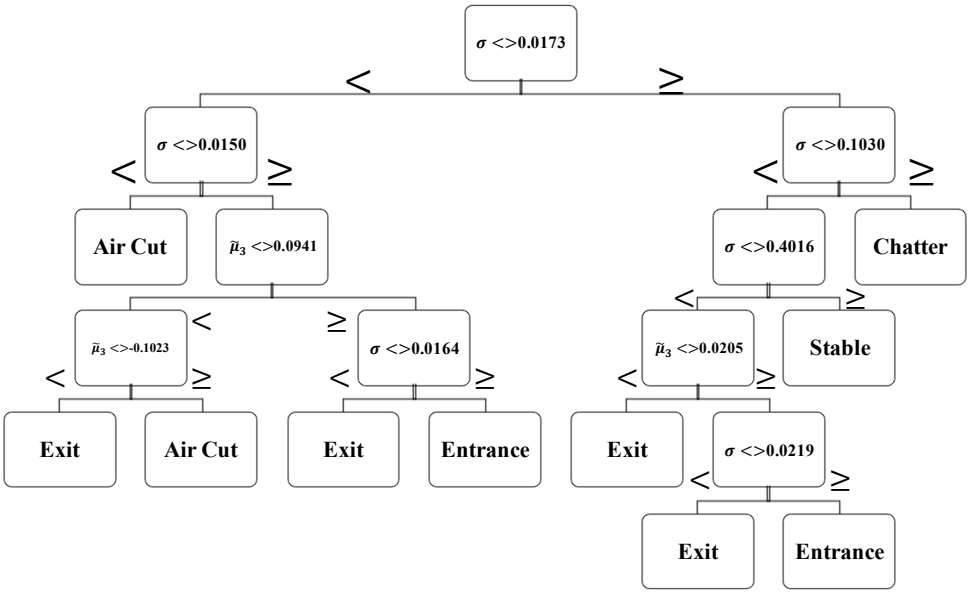

Figure 37 Decision Tree, Decision Criteria for machining state detection

The resultant tree indicates that the chatter signal magnitude tends to have more dispersion from the mean. It shows that the skewness is a good differentiator between the machining states. Meanwhile, the standard deviation is a better discriminator for chatter versus stable and out of the cut.

In order to depict the decision boundaries of the trained decision tree, Figure 38 is drawn. The algorithm detects chatter in most cases when the standard deviation is higher than 0.103, and if it is equal or smaller than 0.04, it detects transient state or air cut. Moreover, it appears that the decision tree is using skewness, mostly, as a discriminator between different non-chatter machining state.

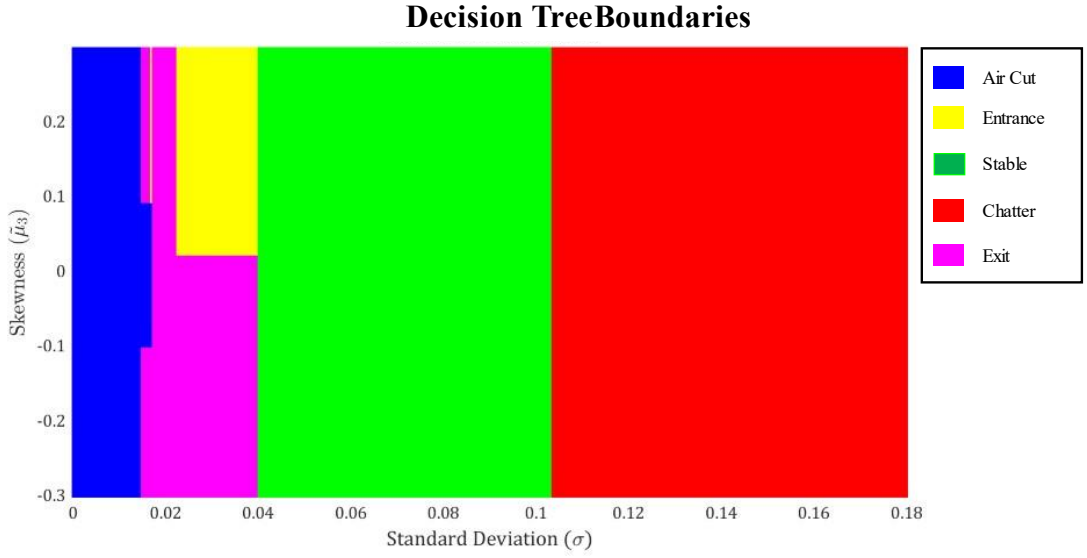

Figure 38 Decision Tree Decision Boundaries

The histogram of each machining state's magnitude for all the training data is shown in Figure 39. The histograms are drawn with 100 bins. A normal distribution is fitted to each histogram. As demonstrated, the stable cut vibration data magnitude’s distribution has a small deviation with a small skewness towards the left, while the chatter cut distribution is farthest from the normal distribution and with a higher skewness. Also, the similar value of standard deviation in air cut,

entrance and exit and the considerably different skewness value in the mentioned machining states supports the findings about the fitted decision tree split strategy.

As evident from histogram of data in Figure 39, and the decision boundaries of the machine learning model given in Figure 38, the main difference between the entrance and exit signal is that the exit signal is mostly right-skewed. In contrast, the entrance signal is significantly left skews in most cases. Moreover, the air cut vibration signal has a small deviation.

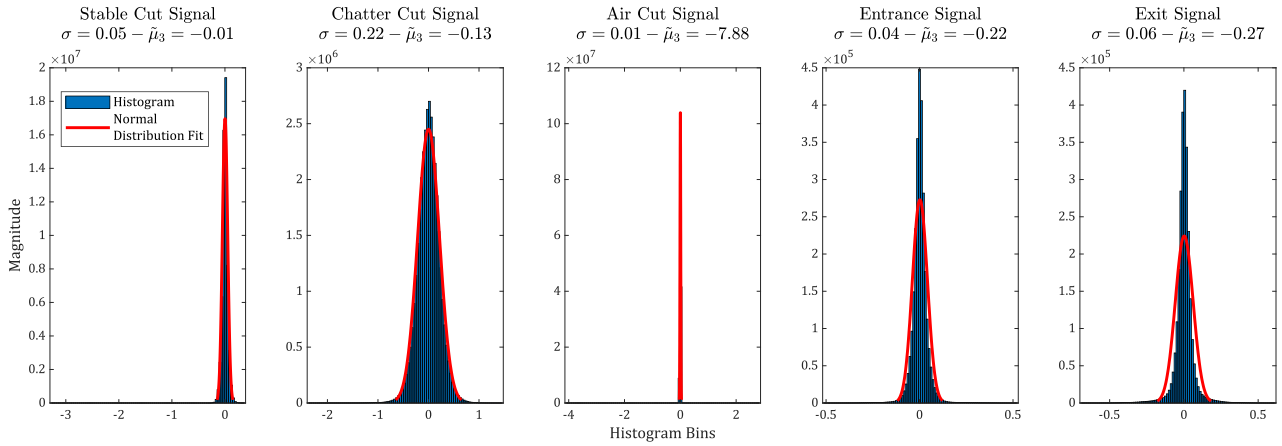

Figure 39 Normal Distribution Fitted to the Histogram of Each Machining State Experiments Time-domain Test Data

Even though the confusion matrix suggests that the model performs poorly on the detection of exit and entrance, the model's overall accuracy is promising and an indicator of the plausibility of solving this problem with machine learning, since it provides reasonable accuracy for real-world implementation. Needless to say, as it is apparent in Figure 36, entrance and exit states are mostly mistaken with air cut and stable, which is harmless for our goal of chatter detection and avoiding the confusion between chatter and transient states of cut.

5.3 Standalone Transfer Learning Machining State Detection Method Performance

This section presents the performance of the proposed standalone machining state detection method introduced in section 3.4. The method is capable of detecting machining state based on the machine's vibration signal.

Table 2 Custom Designed Machining State Detection Model Training Hyperparameters

Hyperparameter	Value
Solver	Adam (Adaptive Moment Estimation) [44]
Initial Learning Rate	$3.0e-4$
Learning Rate Drop Factor	0.08
Denominator Offset (ϵ)	$1.0e-8$
Gradient Decay Factor (β_1)	0.9
Squared Gradient Decay Factor (β_2)	0.999
L2 Regularization(λ)	$1.0e-4$
Mini Batch Size	256
Number of Epochs	30

The learning rate is the step size for updating the learnable parameters of the machine learning model in each iteration. The learning rate has been multiplied by the Learning Rate Drop Factor after each epoch to fine-tune the machine learning model. It results in searching for a wider space at the beginning to find an estimation of the local minimum area; as it proceeds with iterations, the learning rate drops, which results in a more refined search.

The learning is multiplied by the Learning Rate Drop Factor after each epoch. The data is shuffled before training to ensure that the order of data does not play a role in the process of machining state detection and the model performance, i.e. it does not overfit. Training of the model takes 60

minutes on the NVIDIA® Quadro® P4000 GPU with 8 gigabytes of GPU memory. The architecture's complexity, i.e. the number of trainable parameters, is equal to 5.7×10^7. The training process is shown in Figure 40.

Figure 40 Baseline Transfer Learning method's Training Process

After the 4th epoch, the validation accuracy remained constant, which indicates that the model found the local minima. It is worthwhile to mention that the validation accuracy and loss are shown in Figure 40 to showcase the model training process.

For a sample given piece of stable cut vibration's frequency-domain signal input, the output of each convolutional layer's filter is shown in Figure 41 to illustrate the progress of input through the trained model,

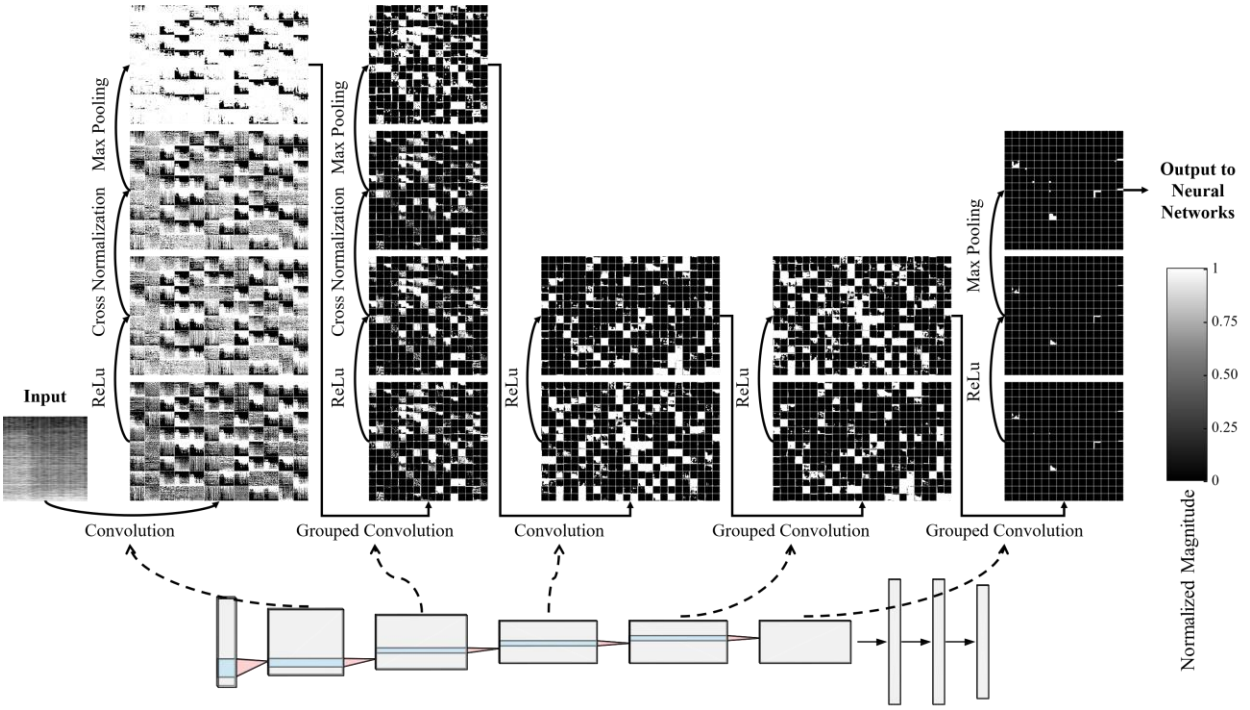

Figure 41 The Output Value of Activation of the CNN Convolutional Layers for a given piece of Stable Frequency-domain Vibration Signal

The trained model has 88.97% test accuracy and 95.24% training accuracy in machining state detection. Each prediction using the trained model requires 0.14 seconds of vibration data and the computation time takes approximately nine milliseconds.

The confusion matrix shown in Figure 42 presents the performance of the transfer learning model. Each row of the matrix is normalized to the true state corresponding to the row; accordingly, the matrix's diagonal cells represent the Recall Value, and the summation of the percentages presented in each row is equal to 100%.

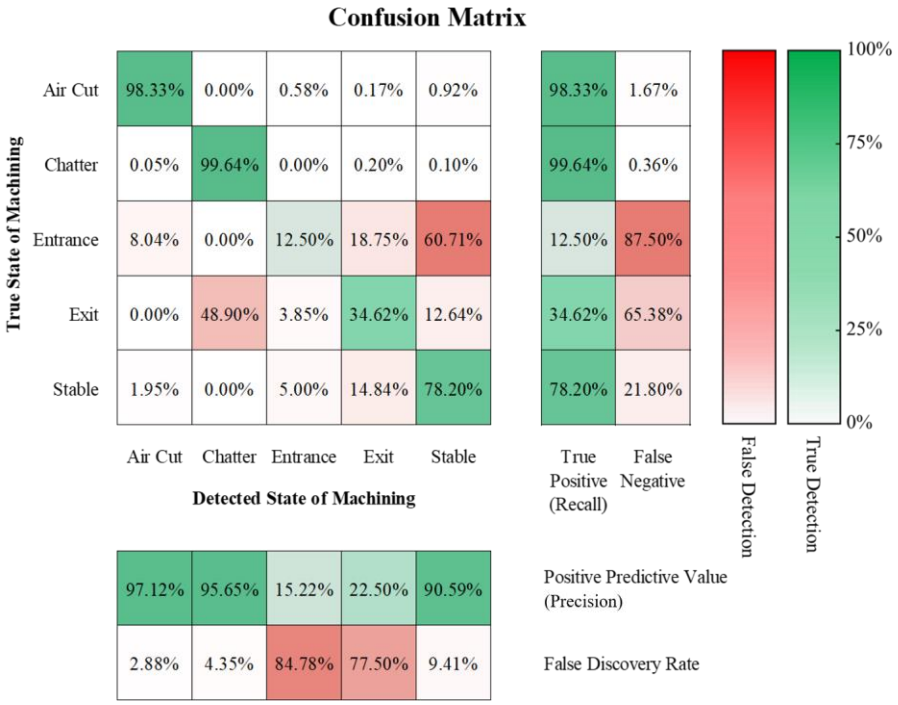

Figure 42 Normalized Confusion Matrix of the Transfer Learning Model

The confusion matrix indicates that the model performs well in detecting chatter cases, though it has a high tendency to misclassify non-chatter cases as chatter. The model performs better than the one suggested in the previous section, both in terms of chatter detection and machining state detection.

5.4 Standalone Custom Designed Convolutional Neural Network Performance

In this section, the process of the machining state detection architecture design suggested in section 3.5 is elaborated, and the numeric results of the proposed method are shown. Moreover, the performance of the mentioned method is evaluated. The results of both architecture design and training of the designed architecture are interpreted in this section.

The architecture design is done through 600 iterations of Bayesian Optimization, searching in search space of 10^{38}, by design, training, and evaluation of one CNN architecture per iteration. The training process is done on an NVIDIA® Quadro® P4000 with 8 gigabytes of GPU memory in 4.2 GPU days.

The architecture design process is shown below.

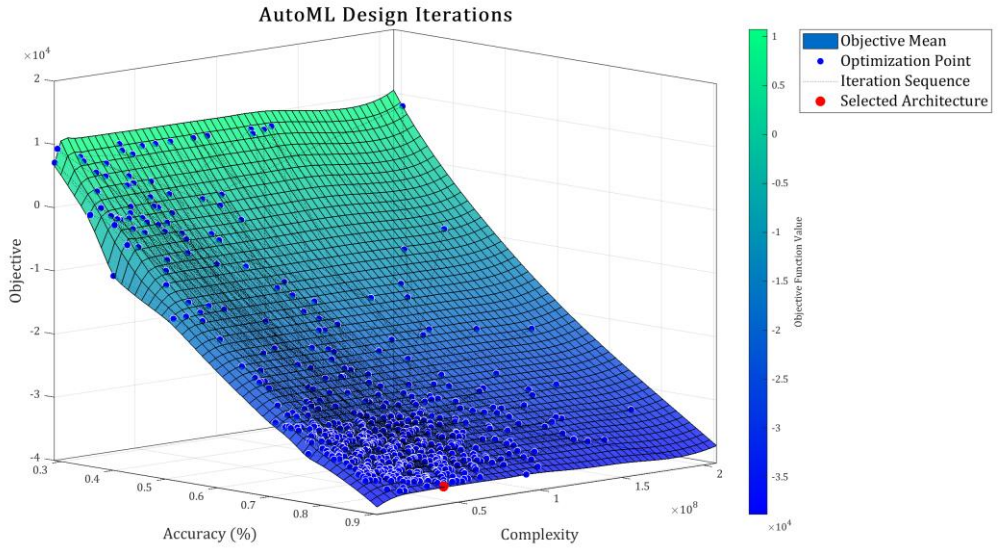

Figure 43 Automated Machine Learning Architecture Design Iterations

The objective function is a combination of complexity and accuracy, as presented by Equation (51). Figure 43 provides another representation of the change of accuracy and complexity during the optimization iterations. It depicts the validation accuracy, and the complexity is compared with the objective function value, middle graph. As shown in Figure 43, the algorithm converges towards points with lower complexity architectural complexity and higher validation accuracy.

Figure 44 depicts the value of the objective function versus Validation accuracy and network complexity in each iteration of the network design process.

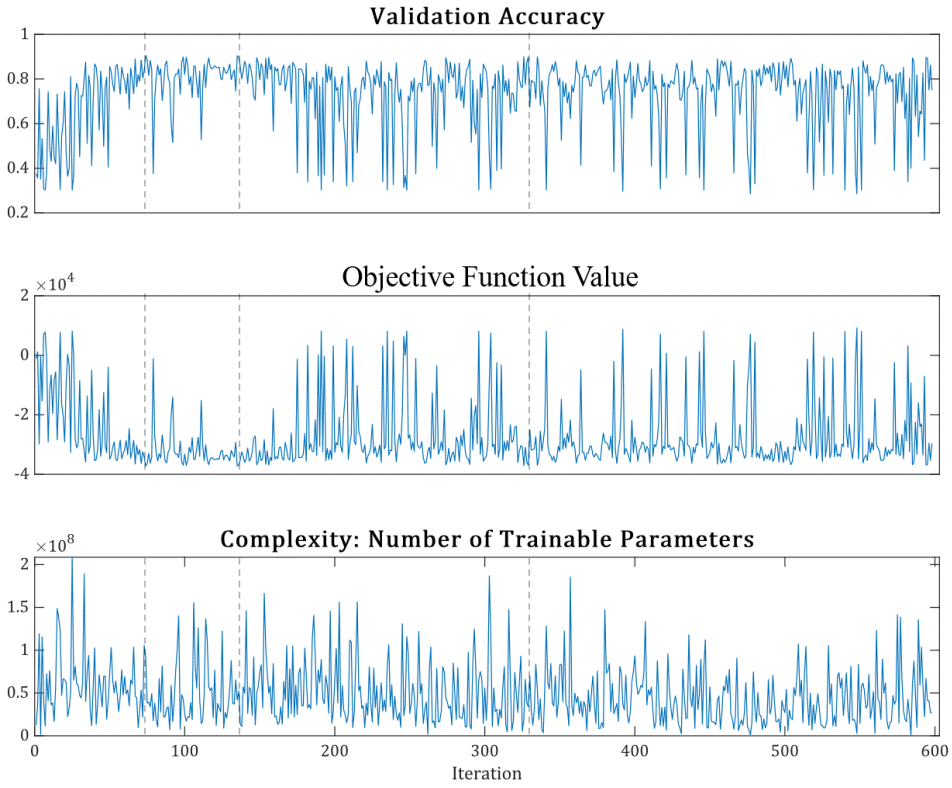

Figure 44 Architecture Validation Accuracy, Objective Function Value, and Complexity of Each Bayesian Optimization Iteration with top three minimum objective function value iterations marked

The estimated objective value by Bayesian optimization versus the real value of the objective function during the optimization is explained as follows,

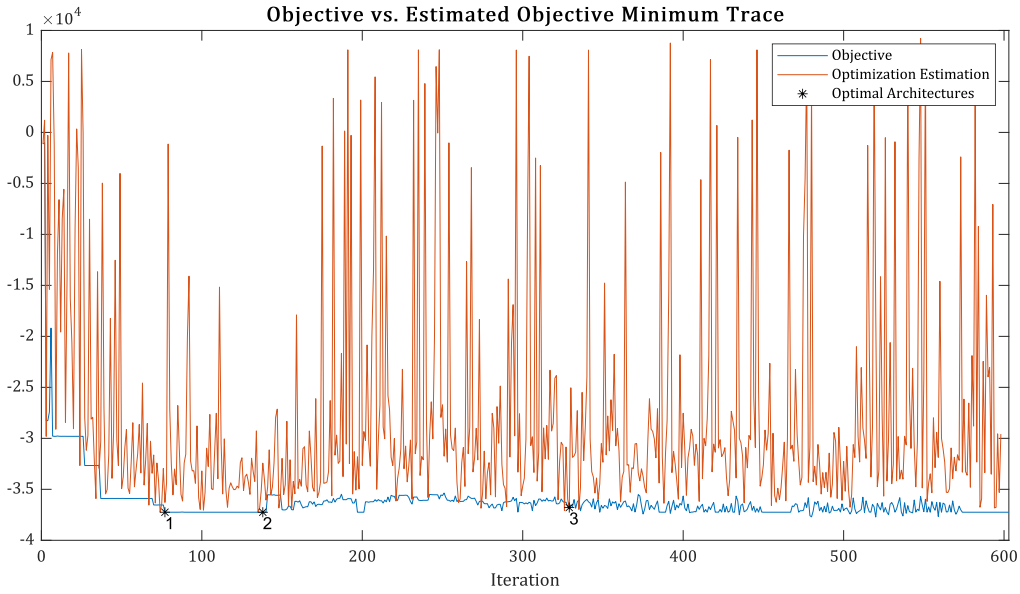

Figure 45 Objective function Value versus Bayesian Optimization Estimation of Objective Function Value During Architecture Design Process

The periodic exploration and exploitation process is apparent in Figure 45. The best objective function values are shown in Figure 45. Despite that the lowest objective function value is determined in the 77th iteration, the second-best architecture resulted in the 138th iteration is used as the final model for machining state detection. The reason is the slightly better performance of the second-best architecture after manual hyperparameter tuning, 0.5% improvement in accuracy, and its extremely low complexity. In other words, after tuning the initial learning rate, and learning rate drop factor, and having it get trained for more number of epochs, the second-best architecture showed a better performance than what the first one shown after fine-tuning.

As shown in Figure 46, the best architecture consists of three convolutional blocks and two nueral network hidden layers. The first block is a set 100, 12×4 convolution kernels with a stride of 4 followed by a ReLu, a cross normalization with a channel size of 5 and $\alpha = 1e - 4$ and $\beta = 0.75$ suceeded by a 3×3 pooling layer with two strides and zero paddings. The second convolution

block consists of 72, 8×3 grouped convolution with two groups and two as paddings, and after that, a ReLu followed by a cross normalization with a channel size of 5 and $\alpha = 1e-4$ and $\beta = 0.75$ suceeded by a 3×3 pooling layer with two strides and zero paddings. The convolutional blocks' output goes into a two layers fully connected neural network followed by a ReLu and a drop out layer. The neural network has 1744 neurons in its hidden layer and five neurons in its output layer, each corresponding to one machining state's probability.

The architecture's complexity, i.e. the number of trainable parameters, is equal to $4.19e7$ which is 31% less complex than the transfer learning method, and the architecture proposed by A. Kerizhevsky [30] while it detects machining states with higher accuracy.

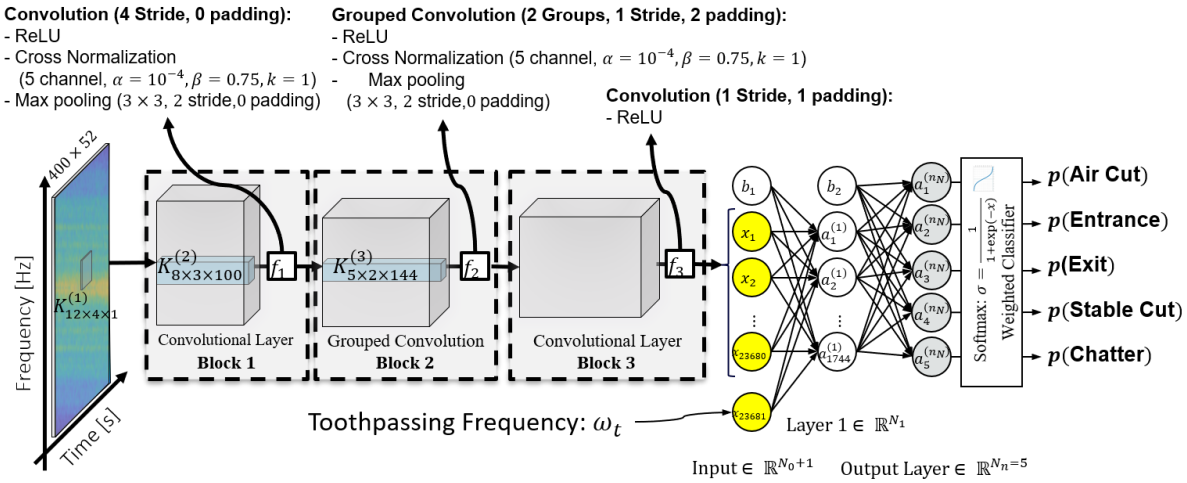

Figure 46 Custom Designed Machining State Detection CNN Architecture Schematic

For a sample given piece of frequency-domain signal input, the output of each convolutional layer's filter is shown below to illustrate the progress of input through the trained model,

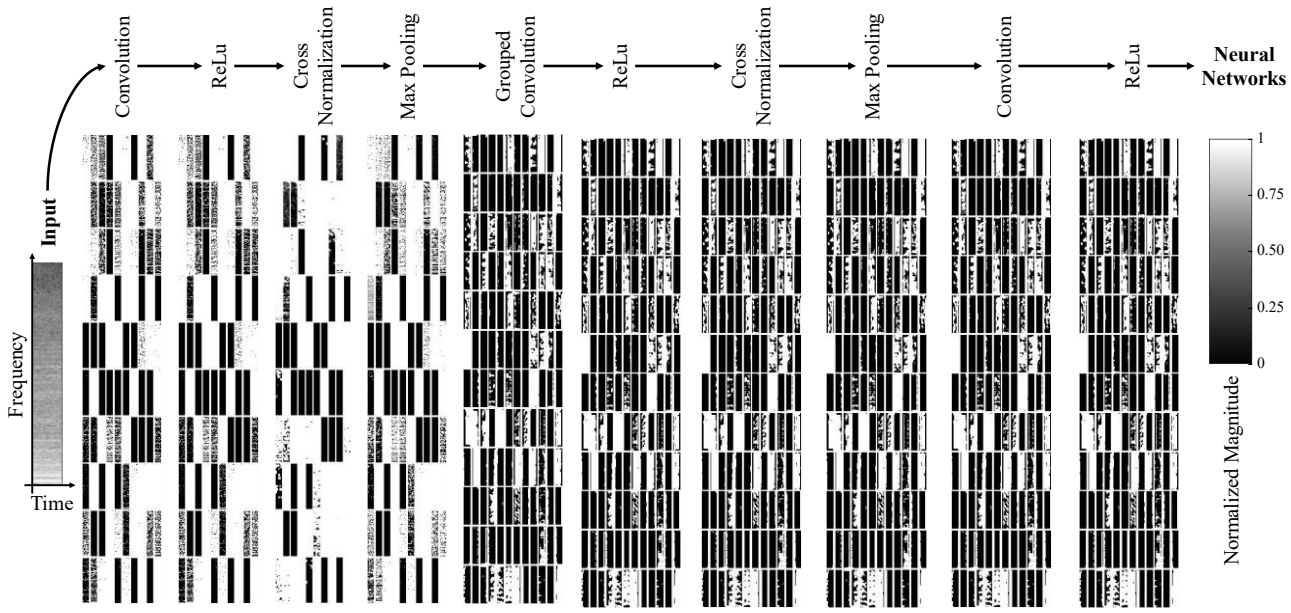

Figure 47 The Output Value of Activations of the CNN Convolutional Layers for a given Piece of Frequency-domain Vibration Signal

After the architecture design process, the training hyperparameters are manually tuned for the designed architecture using the validation data. The resultant model is trained using the following training hyperparameters shown in Table 3,

Table 3 Custom Designed Machining State Detection Model Training Hyperparameters

Hyperparameter	Value
Solver	Adam (Adaptive Moment Estimation) [44]
Initial Learning Rate	During Optimization: $5.0e-5$
	During Training: $3.0e-4$
Learning Rate Drop Factor	During Optimization: 0.05
	During Training: 0.67
Denominator Offset (ϵ)	$1.0e-8$
Gradient Decay Factor (β_1)	0.9

Squared Gradient Decay Factor (β_2)	0.999
L2 Regularization(λ)	$1.0e-4$
Mini Batch Size	512
Number of Epochs	During Optimization: 3
	During Training: 30

The learning is multiplied by the Learning Rate Drop Factor after each epoch. The data gets shuffled before training to ensure that the order of data does not play a role in the process of machining state detection and the model performance, i.e. it does not overfit. Training of the model takes 55 minutes on the GPU mentioned above. This network's training time is roughly equal to the training time of the transfer learning method because of its smaller size. The training process is shown in Figure 48,

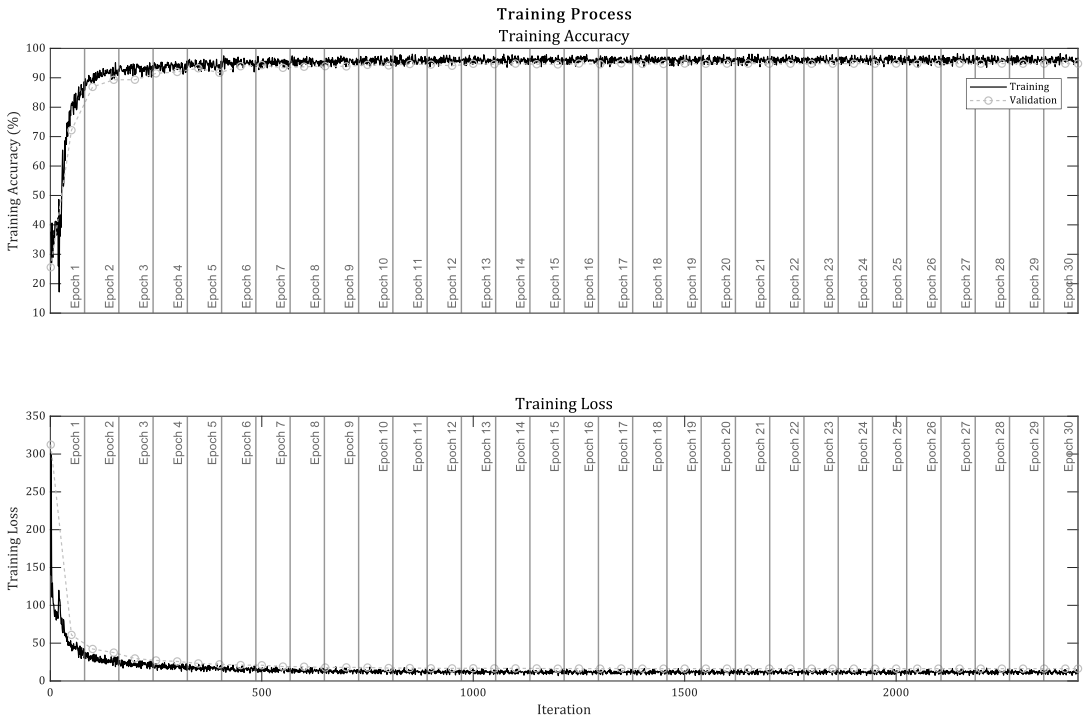

Figure 48 Custom Designed Architecture Training Process

The validation dataset is merely used during the architecture design process to evaluate each designed and trained network's performance. The validation data is not used during the training process, yet, it is used in Figure 48 for illustration.

The designed architecture showed 94.28% test accuracy, 94.81% validation accuracy, and 96.64% training accuracy in the detection of machining states. By disregarding the machining states and using the system merely for cutting stability detection, i.e. considering any state other than chatter as stable, the trained model's test accuracy is equal to 98.88%.

Each prediction using the trained model requires 0.07 seconds of vibration data and the computation time takes approximately three milliseconds. The trained model occupies 154 MB on the memory to store its weights and biases. The closeness of training, validation, and test accuracy

suggests that the model is well trained and not overfitted. It is worthwhile to mention that the validation accuracy and loss are shown in Figure 48 merely to showcase the training process.

The model has 5.31% more accuracy in machining state detection in comparison with the transfer learning model, while it is 31% less complex and provides faster training and three times faster prediction.

The confusion matrix of the trained model is presented in Figure 49 based on the test data, with each row normalized,

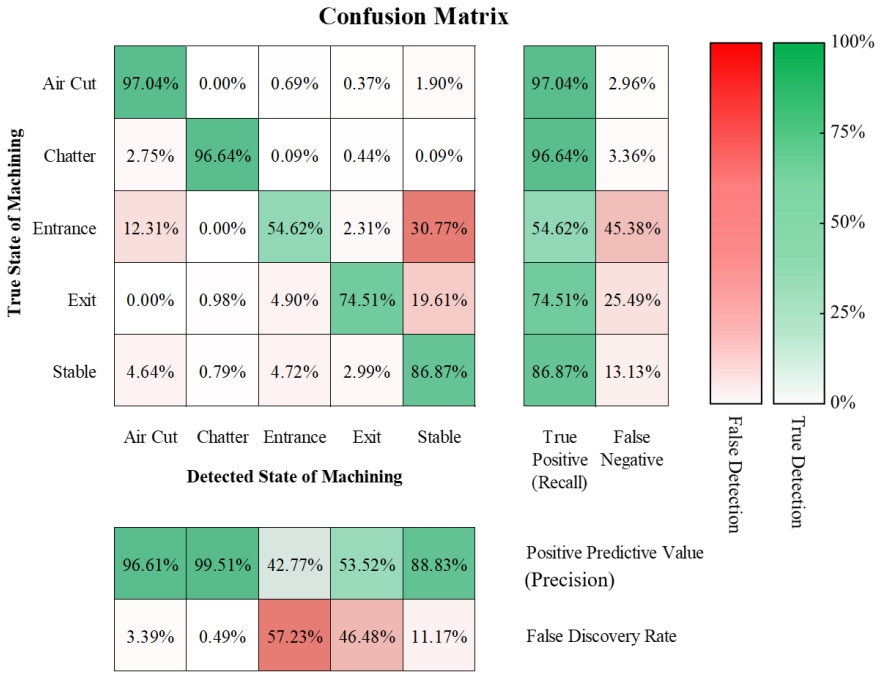

Figure 49 Trained Model of Custom Designed Architecture's Confusion Matrix

The trained model has great dexterity in differentiating chatter, and stable cuts as the percentage of the chatter cases being detected as stable is 0.09% and the accumulative stable and chatter confusion percentage is less than 0.1%. Moreover, it displays that the model is capable of detecting the transient state with a reasonable degree of certainty. The most confusions are in differentiation between the transient states and stable cut, which is justifiable by the fact that there are similar

significances in the stable cut frequency-domain signal, such as tooth passing frequency magnitude, which presents in both transient input and the stable cut. Additionally, in this thesis, the machining states are used to compensate for the energy-based method incapability in distinguishing chatter and transient tool entrance and exit states, which means mistaking transient state with stable cut does not cause disruption in the hybrid system. The precision, recall, and F1-Score of the model for each machining state are given in the below table. The below values indicate the high preciseness and robustness of the suggested model in the detection of cutting states and air cut, according to the F1-score.

Table 4 Custom Designed Machining State Detection Model Performance tested on the Test Dataset

Performance Metric	Air Cut	Chatter	Entrance	Exit	Stable
Recall	90.04%	96.64%	54.62%	74.51%	88.87%
Precision	96.61%	99.51%	42.77%	53.52%	88.83%
F1-Score	0.968	0.981	0.705	0.623	0.878

5.5 Improved Energy-based Chatter Detection and Hybrid System (Machine Learning Fusion) Performance

This section presents the results and experimental validation of the physics-driven improved energy-based method, introduced in section 4.1 and the combination of it with the machine learning method introduced in 3.5, i.e. the hybrid method introduced in section 4.2.

The improved energy-based chatter detection system suggested in section 4.1 results in a chatter detection system with 93.51% accuracy. The confusion matrix of the mentioned model is presented below.

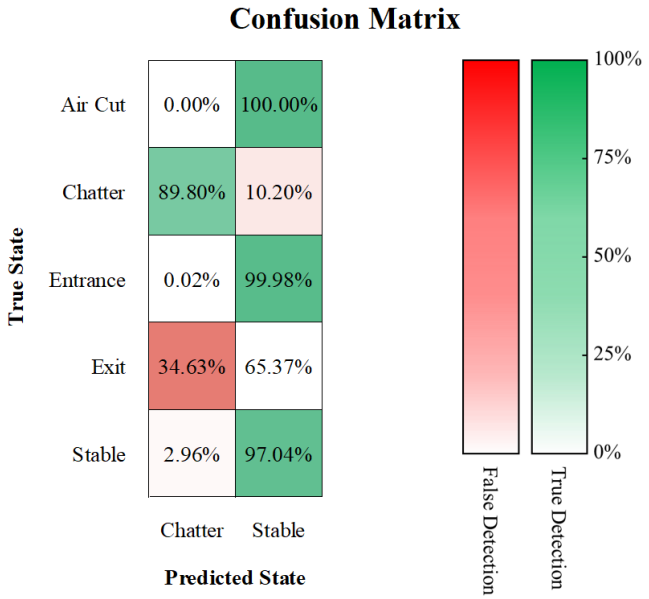

Figure 50 Improved Energy-based Chatter Detection Confusion Matrix with Normalized Rows, based upon the Test Data

The presented confusion matrix is normalized to its rows. The energy-based system is merely capable of detecting chatter and it is desired to have all other machining states be detected as stable. Hence, the predictions are limited to chatter and stable cuts. In this case, it is desired to detect any machining state other than chatter as stable. The confusion matrix's colour coded accordingly. The improved energy-based method has an accuracy of 93.51% in chatter detection. It is 33.93% more accurate than the original energy-based model, proposed by Caliskan [45]. It shows considerable improvement in the misclassification of entrance and exit as chatter, 77.36% and 17.03% improvement, respectively. Besides, the proposed model has a 56.51% improvement in detecting chatter.

Combining the machine learning system and energy-based chatter detection is fused using the method suggested in section 4.2.1. The mentioned system is implemented with $th_{\text{Certainty}} = 0.8$

and $K_p = 0.3$. It resulted in a machining state detection with 94.26% test accuracy in machining state detection and 98.90% in chatter detection. The hybrid model shows 5.39% more accuracy than the standalone improved energy-based chatter detection. Moreover, it performs slightly more accurate (0.02%) than the stand-alone custom CNN architecture. However, its primary importance is its higher robustness, given the fact that it is infusing the data-driven method with the physics of milling. The row normalized the confusion matrix of the hybrid method is presented below,

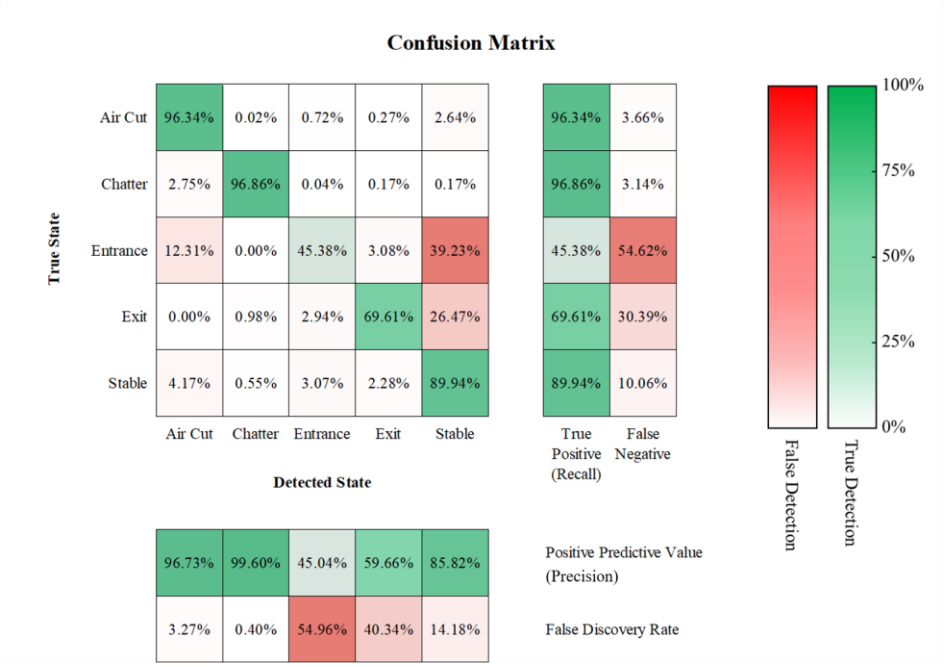

Figure 51 Hybrid (Machine Learning Fusion) Chatter Detection System Confusion Matrix with Normalized Rows

The confusion matrix shows that the model rarely mistakes the transient states with Chatter. The most confusion is in stable cut and transient, entrance and exit, states. It does not affect the chatter detection and suppression process. Moreover, the high confusion percentage of entrance and exit instances with air cut and stable is justifiable because the corresponding data to entrance and exit both include the instances with the tool having slight engagement with the workpiece, i.e. the first instances of entrance and last instances of exit, and the tool being almost fully engaged with the

workpiece, i.e. last instances of the entrance and first instances of exit. The mentioned cases make some of the entrance and exit instances hard to differentiate from air cut and stable cut. The F1-scores for air cut, chatter, entrance, exit, and stable are 0.97, 0.98, 0.62, 0.64, and 0.88.

5.6 Summary

The training and validation of the models presented in sections 3.2.2, 3.4, 3.5, and 4.2 are shown in this chapter. The accuracy of the baseline machine learning method suggested that it is plausible to solve the problem of online machining state detection using data-driven methods. The transfer learning model showed higher accuracy than the baseline model. The other significance of this model is its three times faster response time. The baseline model's network architecture provided the initial state of Bayesian optimization used to design and fabricate the custom CNN architecture. The designed architecture showed a good performance both in chatter detection and machining state detection.

The online physics-driven chatter detection method suggested in 4.2 is validated in this chapter using experimental data. Furthermore, the final system, which consists of the custom CNN architecture and the proposed energy-based chatter detection method, is validated.

It is shown that the custom automated designed machine learning architecture and the hybrid model are essentially showing the same results both in terms of chatter detection and machining state detection. Nevertheless, the hybrid model offers more robustness due to accounting for milling physics, which adds significant value to the design.

Moreover, it provides a reliable platform for the machine learning model to evolve when it comes to low certainty detections. The small lead of the hybrid system in chatter detection and the small

lead of the custom CNN model in the machining state detection is due to the fact that the physics-driven method is merely designed with the goal of chatter detection in mind.

Chapter 6: Conclusion

This research aimed to detect the machining states and suppress instability in the machining process online. The approach taken towards differentiating between the five states of machining, air cut, entrance, stable cut, chatter, and exit, is to amalgamate a data-driven machine learning method with the machining process's physics in order to detect chatter without compromising the robustness. The outcome of this work is a self-evolving machining state detection system. The data-driven models are trained on a vast, diverse set of data, and the performance of each individual system, as well as the final product, are evaluated and reported based upon an untouched collection of experiments with both similar and different cutting conditions.

This work consists of three machine learning models, each capable of detecting machining states as well as cutting instability. The first suggested model provides a simple data-driven solution to machining state detection based on the time-domain vibration signal’s dispersion and skewness and using decision trees. The model shows 85.46% accuracy in machining state detection and 97.11% accuracy in chatter detection. The model is used merely as a proof of concept and an indicator of the possibility of solving machining state detection problems with data-driven methods.

Furthermore, a transfer learning approach is taken towards machining state detection using machine learning. The method uses a CNN architecture introduced by A. Krizhevsky[30] and modifies and partially retrains the architecture on the machining frequency-domain signal. A pipeline is designed to prepare the data and direct the data flow. The pipeline encapsulates filtering, conversion to the frequency domain, using STFT, and training, as well as utilization of the trained

model for machining state detection. The trained model has an 88.97% accuracy in machining state detection and 97.97% accuracy in chatter detection.

As the final deep learning model, an automated machine learning architecture design scheme is created to construct a CNN architecture specialized in machining state detection. The scheme uses Bayesian optimization to tweak the network parameters with the goal of lowering the complexity and increasing the accuracy of the model in the machining state detection. The custom-designed architecture takes the frequency-domain machining vibration signal as well as the tooth passing frequency as its input and outputs the probability of each state of cut. It uses the same pipeline for its data flow as the one designed for the transfer learning method, except it does not require resizing the input. The method developed an architecture providing higher accuracy than the transfer learning architecture, while it has a lower complexity. The trained model showed 94.28% accuracy on machining state detection and 98.88% accuracy in chatter detection, based on examination on the test dataset.

The physics-driven online chatter detection method is based upon online energy-based chatter detection introduced by Caliskan [11]. The method is improved by suggesting a new approach to calculate the chatter energy. In this work, the energy-based chatter detection is tuned, and its chatter energy estimation module using Teager-Kaiser Nonlinear Energy Operator is replaced. The new proposed chatter energy estimation method uses the statistical measures of the FFT of the non-periodic parts of the vibration signal alongside the tooth passing frequency to provide a robust estimation of the chatter energy. The designed system shows 93.51% accuracy in chatter detection with 43.64% improvement over Caliskan's algorithm while minimizing the false detection caused by air cuts.

Both outputs of CNN custom architecture and the enhanced energy-based models are considered together through a decision-making algorithm to form a hybrid machining state detection and chatter suppression system. The hybrid system has 94.26% accuracy in machining state detection and 98.90% accuracy in chatter detection.

The system gets evolved by retraining the machine learning model based on its low certainty detections that have high accumulative probability measured by the hybrid system judgment.

6.1 Future Work

The research can be extended by having a central machine learning model shared among multiple machine tools. Given the small size of the trained machine learning model and its self-evolution capability, it could be shared among multiple machine tools in an industry 4.0 setup. For each retraining instance, the corresponding signal can be uploaded to the server and queued to retrain the model. Then other machines in the setup can download the updated model from the server to have a model with higher accuracy and avoid repetition of the same issue on other machines. The interconnected machine setup could be both local or connected through the internet.

Additionally, a machine learning model could be trained to perform online chatter prediction. It could predict chatter by being trained on the transient entrance signal that results in a chatter cut, as well as the transient signal when the stable cuts start to become unstable, for instance, in slope cuts, i.e. ramping operation, when the depth of cut increases to the point of chattering.

In terms of the chatter detection algorithm, the model could be pre-trained on simulated data before training it on the experiment data. The primary constraint is to provide simulated data for the transient entrance and exit states as well as the chatter signal.

Furthermore, time-domain series machine learning methods, such as Long Short-term Memory (LSTM) methods, could be considered to be replaced with CNNs recommended in this thesis.

Moreover, after chatter detection, the same stability lobes from the beginning of the machining process is used for determining the new stable spindle speed. It is suggested to update the stability lobes in case of encountering chatter and before choosing the new spindle speed based on it.

Bibliography

[1] Y. Altintas, G. Stepan, E. Budak, T. Schmitz, and Z. M. Kilic, “Chatter Stability of Machining Operations,” *J. Manuf. Sci. Eng.*, vol. 142, no. November, pp. 1–46, 2020.

[2] E. Budak and Y. Altintas,, “Analytical Prediction of Chatter Stability in Milling—Part I: General Formulation,” *J. Dyn. Syst. Meas. Control*, vol. 120, no. 1, pp. 22–30, Mar. 1998.

[3] Altintas Y , Author and Ber AA , Reviewer, “Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design,” *Appl. Mech. Rev.*, vol. 54, no. 5, pp. B84–B84, Sep. 2001.

[4] T. Insperger, B. P. Mann, T. Surmann, and G. Stépán, “On the chatter frequencies of milling processes with runout,” *Int. J. Mach. Tools Manuf.*, vol. 48, no. 10, pp. 1081–1089, 2008.

[5] T. Insperger, G. Stépán, P. V Bayly, and B. P. Mann, “Multiple chatter frequencies in milling processes,” *J. Sound Vib.*, vol. 262, no. 2, pp. 333–345, 2003.

[6] Y. Altintas, G. Stepan, D. Merdol, and Z. Dombovari, “Chatter stability of milling in frequency and discrete time domain,” *CIRP J. Manuf. Sci. Technol.*, vol. 1, no. 1, pp. 35–44, 2008.

[7] J. TLUSTY, “CHAPTER 2 - THE THEORY OF CHATTER AND STABILITY ANALYSIS,” in *Machine Tool Structures*, F. KOENIGSBERGER and J. TLUSTY, Eds. Pergamon, 1970, pp. 133–177.

[8] T. Delio, J. Tlusty, and S. Smith, “Use of audio signals for chatter detection and control,” *J. Manuf. Sci. Eng. Trans. ASME*, vol. 114, no. 2, pp. 146–157, 1992.

[9] T. Choi and Y. C. Shin, “On-line chatter detection using wavelet-based parameter estimation,” *J. Manuf. Sci. Eng. Trans. ASME*, vol. 125, no. 1, pp. 21–28, 2003.

[10] M. Lamraoui, M. Thomas, and M. El Badaoui, “Cyclostationarity approach for monitoring chatter and tool wear in high speed milling,” *Mech. Syst. Signal Process.*, vol. 44, no. 1, pp. 177–198, 2014.

[11] H. Caliskan, Z. M. Kilic, and Y. Altintas, “On-Line Energy-Based Milling Chatter Detection,” vol. 140, no. November 2018, pp. 1–12, 2019.

[12] R. G. Brown, P. Y. C. Hwang, and others, *Introduction to random signals and applied Kalman filtering*, vol. 3. Wiley New York, 1992.

[13] J. F. Kaiser, “On a simple algorithm to calculate the ‘energy’ of a signal,” in *International Conference on Acoustics, Speech, and Signal Processing*, 1990, pp. 381–384 vol.1.

[14] P. Maragos, S. Member, J. F. Kaiser, and T. F. Quatieri, “Application to Speech Analysis S :, s :,” *October*, vol. 41, no. 10, pp. 3024–3051, 1993.

[15] W. Lin, C. Hamilton, and P. Chitrapu, “Generalization to the Teager-Kaiser energy function & application to resolving two closely-spaced tones,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1995, vol. 3, no. 3, pp. 1637–1640.

[16] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[17] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[18] M. Postel, B. Bugdayci, and K. Wegener, “Ensemble transfer learning for refining stability predictions in milling using experimental stability states,” *Int. J. Adv. Manuf. Technol.*, vol. 107, no. 9–10, pp. 4123–4139, 2020.

[19] M. Lamraoui, M. Barakat, M. Thomas, and M. El Badaoui, “Chatter detection in milling machines by neural network classification and feature selection,” *JVC/Journal Vib.*

Control, vol. 21, no. 7, pp. 1251–1266, 2015.

[20] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-Art," *arXiv*, no. Dl, 2019.

[21] C. White, R. Ai, and W. Neiswanger, "Neural Architecture Search via Bayesian Optimization with a Neural Network Model," no. NeurIPS, pp. 1–11, 2019.

[22] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," 2005.

[23] P. P. Mitra and D. P. Schmidt, "On the Effectiveness of Bayesian AutoML methods for Physics Emulators," no. October, pp. 1–11, 2020.

[24] I. Rebai, Y. BenAyed, W. Mahdi, and J.-P. Lorré, "Improving speech recognition using data augmentation and acoustic model fusion," *Procedia Comput. Sci.*, vol. 112, pp. 316–322, 2017.

[25] M. K. Nezami, "Performance assessment of baseband algorithms for direct conversion tactical software defined receivers: I/Q imbalance correction, image rejection, DC removal, and channelization," in *MILCOM 2002. Proceedings*, 2002, vol. 1, pp. 369–376.

[26] A. Celisse, "Optimal cross-validation in density estimation with the L2-loss," *Ann. Stat.*, vol. 42, no. 5, pp. 1879–1910, 2014.

[27] M. Stojćev, "Digital Signal Processing: A Computer Based Approach. Sanjit K. Mitra. McGraw-Hill, New York, 1998, hardcover, 864pp., £68.99 ISBN 0-07-042953-7," *Microelectronics Journal*, vol. 30, no. 8. p. 807, 1999.

[28] A. V Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[29] J. B. ALLEN, "Short Term Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform," *IEEE Trans. Acoust.*, vol. 25, no. 3, pp. 235–238, 1977.

[30] A. (University of T. Krizhevsky, I. (University of T. Sutskever, and G. (University of T. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Curran Assoc. Inc.*, pp. 1097–1105, 2012.

[31] N. FUJIMOTO, T. YUMIBAYASHI, M. ONOUE, M. SUGAMOTO, and A. SUGAMOTO, “Surround Inhibition Mechanism by Deep Learning,” *Proc. Annu. Conf. JSAI*, vol. 2019, no. Jsai 2019, pp. 2H3J201-2H3J201, 2019.

[32] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, 2020.

[33] M. S. Abdelfattah, Ł. Dudziak, T. Chau, R. Lee, H. Kim, and N. D. Lane, “Best of Both Worlds: AutoML Codesign of a CNN and its Hardware Accelerator,” 2020.

[34] M. A. Gelbart, J. Snoek, and R. P. Adams, “Bayesian optimization with unknown constraints,” *Uncertain. Artif. Intell. - Proc. 30th Conf. UAI 2014*, pp. 250–259, 2014.

[35] K. J. Swersky, “Improving Bayesian Optimization for Machine Learning using Expert Priors,” 2017.

[36] S. Journal, R. Statistical, and S. Series, “The Determination of the Order of an Autoregression Author (s): E . J . Hannan and B . G . Quinn Published by : Blackwell Publishing for the Royal Statistical Society Stable URL : http://www.jstor.org/stable/2985032,” vol. 41, no. 2, pp. 190–195, 2009.

[37] G. Claeskens and N. L. Hjort, “A comparison of some selection methods,” in *Model Selection and Model Averaging*, Cambridge University Press, 2008, pp. 99–116.

[38] E. Brochu, V. M. Cora, and N. de Freitas, “A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning,” 2010.

[39] H. M. Gutmann, “A Radial Basis Function Method for Global Optimization,” *J. Glob. Optim.*, vol. 19, no. 3, pp. 201–227, 2001.

[40] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959.

[41] Z. Yao, D. Mei, and Z. Chen, “On-line chatter detection and identification based on wavelet and support vector machine,” *J. Mater. Process. Technol.*, vol. 210, no. 5, pp. 713–719, 2010.

[42] N. Fazakis, S. Karlos, S. Kotsiantis, and K. Sgarbas, “Self-Trained LMT for Semisupervised Learning,” *Comput. Intell. Neurosci.*, vol. 2016, p. 3057481, 2016.

[43] L. Jost, “Entropy and diversity,” *Oikos*, vol. 113, no. 2, pp. 363–375, 2006.

[44] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv Prepr. arXiv1412.6980*, 2014.

[45] H. Caliskan, Z. M. Kilic, and Y. Altintas, “On-line energy-based milling chatter detection,” *J. Manuf. Sci. Eng. Trans. ASME*, vol. 140, no. 11, pp. 1–12, 2018.