

POLITECNICO DI TORINO

Department of Management and Production Engineering

Master Degree in Engineering and Management



**PHYSICS INFORMED NEURAL NETWORKS IN
MANUFACTURING**

Supervisor:

Prof. Giulia Bruno

Candidate:

Giosuè Coppola

Academic Year 2024-2025

Index

Index.....	2
List of Figures.....	4
List of Tables.....	6
Abstract.....	7
Introduction.....	8
Aim of the Thesis.....	8
Structure of the Thesis	9
1. Neural Networks and PINNs.....	10
1.1 ANNs: Artificial Neural Networks	10
1.1.1 Training of Neural Networks	13
1.2 From Neural Networks to Deep Learning.....	14
1.3 Introduction to Physics-Informed Neural Networks (PINNs).....	17
1.3.1 Mathematical and Computational Foundations of PINNs	19
1.3.2 The role of automatic differentiation	20
1.3.3 Challenges and Advances in Training PINNs	20
1.3.4 The Role of Parallelization and High-Performance Computing	23
2. Milling Process.....	25
2.1 Principles of Milling	25
2.2 Types of Milling Operations.....	26
2.3 Equipment and Tooling	26
2.4 Parameters Influencing Milling	27
2.5 Tool Wear	28
2.5.1 Taylor Equation	28
3. State of the Art	29
3.1 Research Process.....	29
3.2 Bibliographic research.....	31
4. PINNs in Milling Process	43

4.1	Experimental method and dataset.....	43
4.1.1	Milling machine and setup	43
4.1.2	The Cutting Process	44
4.1.3	Dataset Composition	45
4.1.4	Tool Wear.....	46
4.2	Physics applied	48
4.3	Python Implementation	49
4.3.1	Libraries	49
4.3.2	Tool Wear Prediction	50
4.3.3	Taylor's parameters prediction	51
4.3.4	Loss Function	52
4.3.5	Data Loss term L_{data} and Mean Square Error on valid indices	53
4.3.6	Initial Condition Penalty L_{ic}	54
4.3.7	Anchor term L_{anchor}	54
4.3.8	Training Loop	54
5.	Results.....	57
5.1	Tool wear prediction DNN	57
5.2	Tool Wear prediction PINN	61
5.3	Results analysis	63
6.	Conclusions.....	68
	Bibliography.....	70

List of Figures

Figure 1- Relationship between AI and Neural Networks [4].....	10
Figure 2 - Visual Representation of FNN, RNN and CNN respectively [5]	12
Figure 3 - Deep Learning Neural Network Architecture [6]	16
Figure 4 - Example of how a Language Generating RNN, combined with a CNN, translates images into descriptive captions by focusing on specific regions [1].	16
Figure 5 - Basic structure of PINN for conservation law [9]	18
Figure 6 - Schematic to illustrate three possible categories of physical problems and associated available data. [9]	18
Figure 7 - The top figure is the schematic of XPINN sub-net employed in a subdomain where NNs and PI part for Burgers equation are shown. The bottom figure shows the irregularity shaped sub-domain division in 'X'-shaped domain.[11].....	23
Figure 8 - Explanation of the difference between a CPU and a GPU [13].....	24
Figure 9 - This image showcases the milling process. The application of a cutting fluid, visible as a spray, serves to cool the tool, reduce friction, and remove chips from the cutting area. [16].....	25
Figure 10 - PINNs reseach on Scopus	29
Figure 11 – PINNs Articles per year	30
Figure 12 - PINNs Articles by subject area	30
Figure 13 - PINNs articles by geographic area.....	30
Figure 14 - PINNs AND Milling Process research on Scopus.....	31
Figure 15 - Matsuura MC-510V [16]	43
Figure 16 - Tool wear VB [28]	44
Figure 17 - shear zones at tool/workpiece interface [28]	45
Figure 18 - Exampe of Tool Wear over time i one of the sixteen cases of the dataset [28]	47
Figure 19 – Tf found as result of the taylor_Tf function and the calculation of the scaled time τ	50
Figure 20 – Fully connected regressor for Flank-Wear prediction	51
Figure 21 – Learnable variables an, am, ap, bk and application of the sigmoind/softplus to them	52
Figure 22 – Loss function as sum of the three loss terms (Ldata, Lic, Lanchor)	53
Figure 23 – Normalization for feed/DOC.....	55
Figure 24 – Sigmoind and Softplus mapping for Taylor’s variables	55

Figure 25 – Per case forward pass	56
Figure 26 – PINNs backpropagation	56
Figure 27 - MLP regressor architecture	58
Figure 28 - DNN Training procedure and optimization	58
Figure 29 - DNN inference pipeline and post-processing	59
Figure 30 - Cases 1-3 DNN VB predictions	59
Figure 31 - Cases 4,5,7 DNN VB predictions	60
Figure 32 - Cases 8-10 DNN VB predictions	60
Figure 33 - Cases 11-13 DNN VB predictions	60
Figure 34 - Cases 14-16 DNN VB predictions	60
Figure 35 - Cases 1-3 PINN VB predictions	61
Figure 36 - Cases 4,5,7 PINN VB predictions	61
Figure 37 - Cases 8-9 PINN VB predictions	62
Figure 38 - Cases 11-13 PINN VB predictions	62
Figure 39 - Cases 14-16 PINN VB predictions	62
Figure 40 - Difference between DNN and PINN MAE for each working condition	64
Figure 41 - Difference between DNN and PINN Variance for each working condition	65
Figure 42 - Percentage of sampling points with error $> 0,05$ mm for DNN and PINN	66
Figure 43 - Percentage of sampling points with error $> 0,10$ mm for DNN and PINN	67

List of Tables

Table 1 - Research papers analyzed for Query 1 34

Table 2 - Research papers analyzed for Query 2 38

Table 3 - Datasets & Physical Laws for Query 2 papers 42

Table 4 - Working conditions of NASA dataset with the relative specifications..... 46

Abstract

This thesis investigates Physics-Informed Neural Networks (PINNs) for predicting flank wear (VB) in milling, targeting scenarios where data are scarce and purely data-driven models struggle to generalize. The proposed PINN embeds an extended Taylor tool-life relationship through a learnable time scaling T_f and re-parameterized exponents (n, m, p) with a positive scale K . The network ingests dimensionless time $\tau = t/T_f$ together with normalized process settings (feed, depth of cut, material) and is trained with a composite objective that combines data fitting on the available VB labels, an initial-condition penalty to enforce near-zero wear at the start, and a physics anchor that nudges the prediction at $\tau = 1$ toward a robust reference wear level. Evaluation follows a leave-one-case-out protocol on the NASA milling dataset (16 operating conditions spanning feed, depth, and material), and results are contrasted against a matched deep neural network (DNN) baseline. The PINN achieves lower average MAE (0.1008 vs. 0.1074), markedly lower error variance (0.0146 vs. 0.0202), and reduced shares of samples exceeding 0.05 mm (61.9% vs. 69.7%) and 0.10 mm (34.3% vs. 38.3%), with degradations largely attributable to isolated first-sample spikes in two cases. Overall, the physics-guided time scaling improves trajectory tracking and stabilizes residuals relative to the purely data-driven baseline, demonstrating a practical route to robust tool-wear monitoring under limited labeled data.

Introduction

In the modern era of manufacturing, the integration of advanced computational tools has become essential to optimize processes and achieve higher efficiency. Among these tools, machine learning (ML) has emerged as a transformative technology, offering capabilities to model complex systems and predict outcomes with remarkable accuracy. However, the success of ML algorithms often depends on the availability of extensive, high-quality data for training. In manufacturing processes, especially in specific operations such as milling, acquiring such data can be costly, time-consuming, or even infeasible. To address this limitation, Physics-Informed Neural Networks (PINNs) have gained attention as a promising alternative.

PINNs are a class of neural networks that incorporate physical laws, expressed as partial differential equations (PDEs) or other domain knowledge, directly into the learning process. By embedding these constraints into the loss function, PINNs enable the model to learn and generalize effectively, even with limited datasets. This innovative approach bridges the gap between traditional physics-based modeling and data-driven methods, making it particularly valuable for applications where data scarcity is a critical concern.

This thesis investigates the application of PINNs within the context of manufacturing, specifically focusing on milling processes. Unlike conventional neural networks, which rely solely on data-driven patterns, PINNs leverage the governing physical principles of the milling process to enhance their predictive accuracy. The overarching goal of this research is to develop a PINN algorithm tailored for milling, demonstrating its potential to overcome data limitations while maintaining robust performance.

Aim of the Thesis

The primary aim of this thesis is to design, implement, and evaluate a Physics-Informed Neural Network for the milling manufacturing process. By embedding physical constraints into the learning framework, the proposed approach seeks to achieve reliable predictions and insights where traditional neural networks may fall short due to insufficient data. This research represents a significant step toward integrating physics-informed methodologies into the domain of manufacturing, offering a potential pathway to more efficient and adaptive process optimization.

Structure of the Thesis

The thesis is organized into six chapters, each addressing a critical aspect of the research:

1. **Chapter 1: Neural Networks and PINNs** – Explains the foundational concepts of Neural Networks and PINNs and highlights the key differences between PINNs and ANNs.
2. **Chapter 2: Milling process** – Presents how a milling process works and explains some of the key characteristics of it.
3. **Chapter 3: State of the Art** – Explain how has been conducted the bibliographic research and presents the main scientific papers used in the development of this thesis.
4. **Chapter 4: PINN in Milling Process** – Details the design and implementation of the PINN algorithm for the milling process, including the underlying physics and computational strategies.
5. **Chapter 5: Results** – Presents the outcomes of the algorithm's application to the milling case study, analyzing its performance and predictive accuracy.
6. **Chapter 6: Conclusions** – Summarizes the findings, discusses implications for the manufacturing industry, and proposes future research directions.

This structured approach ensures a comprehensive exploration of the potential of PINNs in manufacturing, offering both theoretical insights and practical contributions. By bridging the gap between physics and machine learning, this research aims to advance the understanding and application of cutting-edge computational methods in industrial settings.

1. Neural Networks and PINNs

Neural networks have emerged as a cornerstone of modern artificial intelligence [Figure 1- Relationship between AI and Neural Networks], offering powerful tools for solving complex problems across a diverse range of fields [1]. Inspired by the architecture of the human brain, these computational models excel in recognizing patterns, learning from data, and making predictions, making them invaluable in domains such as image recognition, natural language processing, and scientific computing [1][7]. Among the innovative advancements in neural network applications are Physics-Informed Neural Networks (PINNs), which integrate data-driven machine learning with the governing principles of physics [2]. By embedding physical laws, such as conservation laws and differential equations, directly into the learning process, PINNs enable accurate modeling and analysis of systems where traditional numerical methods struggle [2][3]. This chapter provides an overview of the fundamental principles of neural networks and introduces the concept of PINNs, setting the stage for their application in manufacturing, a domain where precision, efficiency, and adaptability are paramount [3].

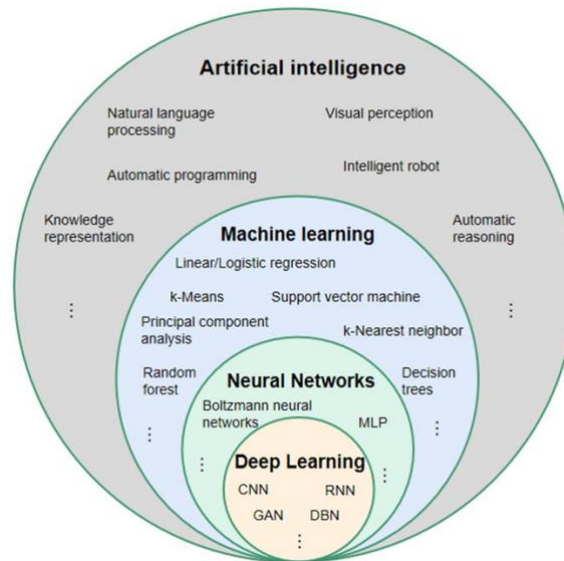


Figure 1- Relationship between AI and Neural Networks [4]

1.1 ANNs: Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational systems inspired by the structure and functioning of biological neural networks in the human brain. They consist of interconnected layers of nodes (neurons), including an input layer, one or more hidden layers, and an output layer, where each connection is associated with a weight that adjusts as the network learns

from data [1]. ANN architectures are broadly categorized based on their structure and functionality:

- **Feedforward Neural Networks (FNNs)** represent the simplest type of artificial neural networks, where data flows in a single direction—from the input layer, through one or more hidden layers, to the output layer. There are no feedback loops or cycles, making them straightforward to design and train. Each layer in an FNN consists of neurons that process information by applying weights and biases, followed by an activation function, such as ReLU, sigmoid, or tanh. FNNs are particularly effective for problems like classification, regression, and pattern recognition. They are foundational in neural network research and serve as the backbone for more complex architectures. However, FNNs lack memory and struggle with sequential or time-dependent data, limiting their applicability in tasks like speech recognition or time-series forecasting [1][8][9][Figure 2].
- **Recurrent Neural Networks (RNNs)** extend the capabilities of FNNs by introducing feedback loops, enabling the network to maintain a memory of previous inputs. This architecture makes RNNs suitable for sequential data, such as time-series analysis, natural language processing, and speech recognition. At each step, the output of a neuron in the hidden layer depends not only on the current input but also on its previous state. This allows RNNs to capture temporal dependencies and contextual relationships within data. Despite their strengths, RNNs face challenges such as vanishing or exploding gradients during training, which can hinder their ability to learn long-term dependencies. To address this, advanced variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) have been developed, incorporating gating mechanisms to manage memory and focus on relevant information over longer sequences [1][8][9][Figure 2 - Visual Representation of FNN, RNN and CNN respectively]
- **Convolutional Neural Networks (CNNs)** are specialized neural network architectures designed for processing structured data, such as images or videos. CNNs utilize convolutional layers that apply learnable filters to detect spatial features like edges, textures, and shapes. These filters slide across the input data, capturing local patterns and hierarchies of features. A CNN typically consists of three key layers: convolutional layers for feature extraction, pooling layers for downsampling and reducing computational complexity, and fully connected layers for decision-making. Activation functions, like ReLU, introduce non-linearity, and dropout

regularization prevents overfitting. CNNs excel in image recognition, object detection, and other computer vision tasks. Architectures like AlexNet, VGGNet, ResNet, and Inception have pushed the boundaries of CNN performance, enabling breakthroughs in applications such as medical imaging, autonomous driving, and video analysis [70 [Figure 2 - Visual Representation of FNN, RNN and CNN respectively].

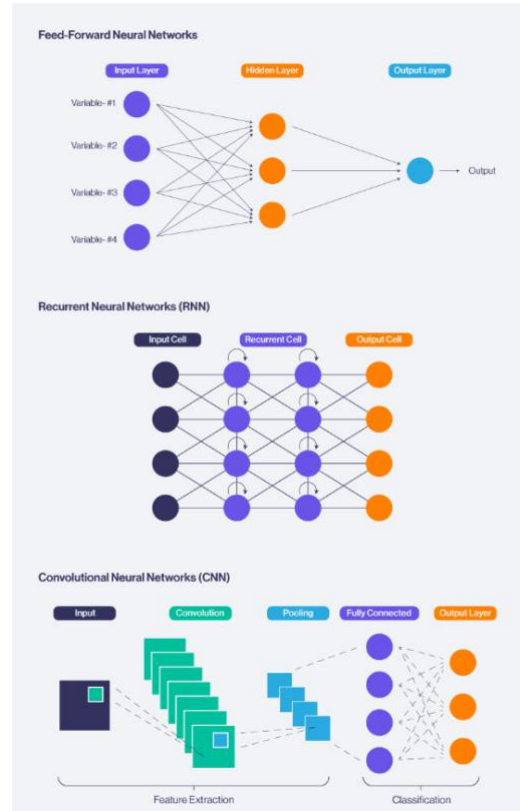


Figure 2 - Visual Representation of FNN, RNN and CNN respectively [5]

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which began in 2010, was a pivotal moment for neural networks. It showcased the ability of CNNs to revolutionize image recognition tasks. In 2012, a CNN model called AlexNet, developed by Krizhevsky, Sutskever and Hinton, achieved unprecedented accuracy on the ImageNet dataset, outperforming traditional machine learning methods and reducing the top-5 error rate by over 10%. This breakthrough not only demonstrated the power of neural networks for real-world applications but also sparked a surge of interest and development in deep learning [9]. Recent advances have also introduced specialized neural networks, such as Support-Vector Networks, which utilize non-linear mappings to construct decision surfaces in high-

dimensional feature spaces for classification tasks, ensuring high generalization ability [10]. Moreover, probabilistic neural networks have been employed in classification tasks by estimating probability density functions and applying Bayesian inference [9]. These versatile architectures have driven breakthroughs in areas such as image and speech recognition, natural language processing, and predictive modeling [1][8].

The rise of deep learning further refined ANN capabilities, enabling the development of deeper architectures capable of learning complex patterns from massive datasets. This advancement paved the way for domain-specific adaptations like Physics-Informed Neural Networks (PINNs), which integrate physical laws (e.g., conservation laws or differential equations) into the neural network framework. PINNs extend traditional ANNs by embedding domain-specific knowledge, making them powerful tools for addressing forward and inverse problems in science and engineering [2][3].

1.1.1 Training of Neural Networks

The training of neural networks is a critical process where the network learns to map input data to desired outputs by adjusting its internal parameters—weights and biases—based on examples from a dataset. This process typically involves three main steps: forward propagation, loss calculation, and backpropagation. During forward propagation, the input data flows through the network layers, where neurons apply weighted sums and activation functions to generate outputs.

The loss function plays a central role in this process, as it quantifies the difference between the network's predictions and the true targets, providing a measure of the model's error[8][9]. Loss functions can be broadly categorized based on the type of task:

- **Mean Squared Error (MSE):** Commonly used for regression tasks, MSE computes the average of the squared differences between predicted and actual values. It penalizes larger errors more heavily, making it sensitive to outliers. The formula is:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- **Cross-Entropy Loss:** Widely used for classification tasks, cross-entropy measures the difference between two probability distributions—the predicted probabilities and the actual class labels. For binary classification, the formula is:

$$L = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

This loss encourages the network to assign high probabilities to the correct class and low probabilities to incorrect classes.

- **Hinge Loss:** Used for classification tasks, particularly in Support Vector Machines (SVMs), hinge loss focuses on maximizing the margin between classes. For a true label y and prediction \hat{y} , the hinge loss is:

$$L = \frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i \hat{y}_i)$$

- **Huber Loss:** A hybrid between MSE and Mean Absolute Error (MAE), Huber loss is robust to outliers and switches between the two loss types based on the size of the error:

$$L = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \leq \delta \\ \delta |y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{for } |y_i - \hat{y}_i| > \delta \end{cases}$$

Once the loss is computed, backpropagation calculates the gradients of the loss function with respect to the network's weights and biases using the chain rule of calculus. These gradients are then used to update the parameters through optimization algorithms like gradient descent or its variants, such as stochastic gradient descent (SGD), RMSprop, or Adam [8][9].

The training process is iterative, often requiring many epochs (complete passes through the dataset) to converge to a solution that minimizes the loss function. Regularization techniques, such as dropout or weight decay, are commonly employed to prevent overfitting, ensuring that the model generalizes well to unseen data [6]. Neural network training is computationally intensive, and advancements in hardware (e.g., GPUs) and software frameworks have significantly accelerated this process, making it feasible to train deep networks with millions of parameters on large datasets. The effectiveness of training largely depends on the choice of hyperparameters, including learning rate, batch size, and network architecture, as well as the quality of the training data [8][9].

1.2 From Neural Networks to Deep Learning

The need for the evolution to deep learning stemmed from the limitations inherent in traditional neural networks, which were unable to efficiently process complex, high-dimensional data. These earlier architectures required extensive manual feature engineering, relying on domain expertise to transform raw data into forms suitable for learning. As tasks in fields such as image recognition, speech transcription, and natural language processing

grew in complexity, the inadequacy of shallow networks became evident. Their limited scalability and inability to model hierarchical features or capture intricate data patterns underscored the necessity for more advanced methodologies. Deep learning emerged as a solution to these challenges, introducing architectures capable of hierarchical feature learning and representation discovery, enabling the direct processing of raw data and significantly improving performance on high-dimensional tasks.

Traditional neural networks, including perceptrons and shallow architectures, relied on handcrafted feature extraction, a process both labor-intensive and constrained by the expertise of engineers. These models struggled to generalize well across diverse datasets, particularly those with high-dimensionality or requiring intricate pattern recognition. Consequently, their application was limited to relatively simple tasks, often failing in domains where data variability and complexity were significant.

Deep learning addressed these shortcomings through the use of deep architectures composed of multiple layers of non-linear processing units [Figure 3 - Deep Learning Neural Network Architecture]. These networks excelled at hierarchical representation learning, where successive layers automatically discovered increasingly abstract features from raw data. For example, in image processing, initial layers might detect edges, intermediate layers identify motifs, and deeper layers recognize objects or scenes. This layered approach allowed deep learning to surpass traditional methods in tasks requiring high-level reasoning.

Deep learning architectures brought several innovations that distinguish them from traditional neural networks:

- **Layered Hierarchies:** By stacking multiple non-linear layers, deep learning systems amplify features essential for discrimination while suppressing irrelevant variations. For instance, in image processing, initial layers detect edges, intermediate layers identify motifs, and deeper layers recognize objects.
- **General-Purpose Learning:** Unlike traditional networks that require specific feature engineering, deep learning models learn directly from data, reducing reliance on domain expertise.
- **Scalability:** Advances in computational hardware, especially GPUs, and large-scale datasets have enabled training of complex deep networks, overcoming limitations of earlier architectures.

- **Breakthroughs Across Domains:** Deep learning has outperformed traditional methods in fields such as drug discovery, genomics, and particle physics, highlighting its versatility and transformative potential.

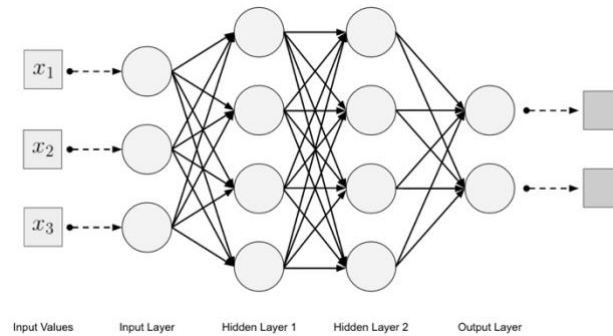


Figure 3 - Deep Learning Neural Network Architecture [6]

Deep learning has consistently achieved state-of-the-art results in numerous areas. CNNs revolutionized image classification [Figure 4], particularly with successes in datasets like ImageNet, while RNNs, and their variants such as long short-term memory (LSTM) networks, have demonstrated exceptional performance in sequential tasks like speech recognition and machine translation. Furthermore, deep learning has proven transformative in scientific discovery, modeling complex systems such as the effects of DNA mutations and the reconstruction of neural circuits.

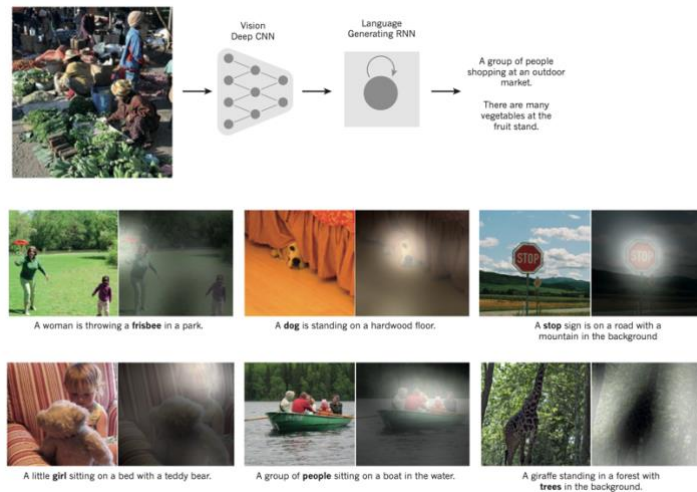


Figure 4 - Example of how a Language Generating RNN, combined with a CNN, translates images into descriptive captions by focusing on specific regions [1].

The general-purpose nature of deep learning ensures its continued applicability as algorithms and architectures evolve. The capacity to scale with data and computational power positions deep learning as a foundational technology, driving innovation across domains ranging from personalized medicine to autonomous systems [1].

1.3 Introduction to Physics-Informed Neural Networks (PINNs)

Over the past few decades, the solution of Partial Differential Equations (PDEs) has played a pivotal role in driving scientific and technological advancements across a wide range of disciplines, including engineering, physics, and applied mathematics. PDEs are the mathematical backbone for modeling complex phenomena such as fluid dynamics, heat transfer, structural mechanics, and quantum mechanics. Traditional numerical methods, such as finite element, finite volume, and finite difference approaches, have long been the cornerstone for solving these equations. While these techniques have proven to be highly effective for many problems, they face significant limitations when confronted with challenges like high-dimensional domains, noisy or incomplete data, and the computational expense of solving large-scale systems with intricate geometries [2][7].

In response to these challenges, **Physics-Informed Neural Networks (PINNs)** have emerged as a transformative paradigm that bridges the gap between machine learning and physics-based modeling. PINNs are a class of deep learning frameworks designed to solve PDEs by embedding the governing equations of physical systems directly into the neural network's architecture. Unlike traditional data-driven neural networks, which rely solely on empirical data for training, PINNs incorporate the laws of physics - expressed as differential equations or other mathematical constraints - as soft penalties in their loss functions. This hybrid approach ensures that the network's predictions are not only data-consistent but also physically meaningful [7][11]. As illustrated in [Figure 5], the basic structure of PINNs involves two neural network components. The first, referred to as the "physics-uninformed" network, approximates the PDE solution $U(x,t)$. The second, the "physics-informed" network, evaluates the residual of the governing PDE $F(x,t)$, ensuring the solution adheres to the specified physical laws. Optimization typically involves minimizing a composite loss function that combines data errors and PDE residuals. The "U" network is often a fully connected Deep Neural Network (DNN), although it can also employ architectures like CNNs, RNNs, or hybrid models. The "F" network's structure is dictated by the PDE itself, with its depth directly proportional to the order of derivatives in the PDE. This architecture makes PINNs flexible and effective for solving a wide range of forward and inverse problems in physics [11].

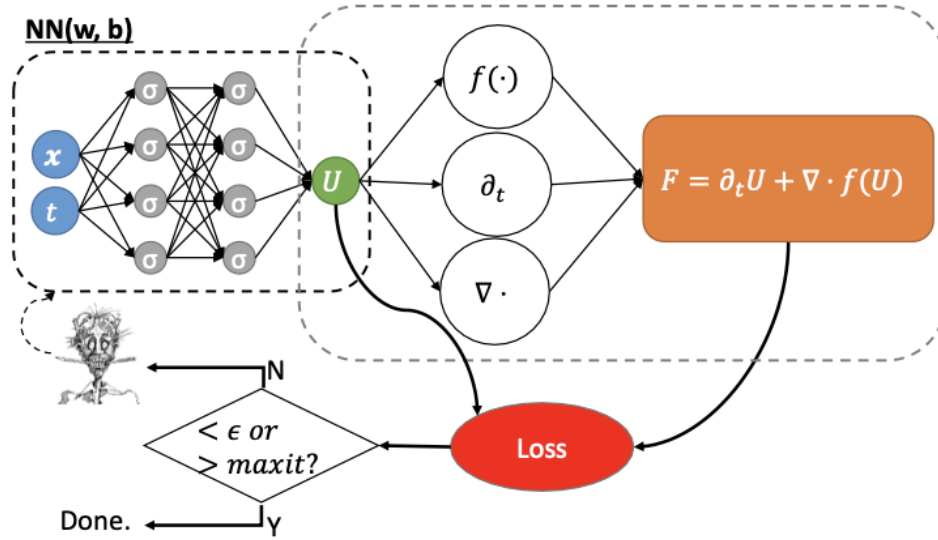


Figure 5 - Basic structure of PINN for conservation law [9]

PINNs are particularly suited for problems where data is sparse, noisy, or expensive to acquire. By leveraging physical laws as priors, PINNs can generalize well even with limited observational data [Figure 6]. For example, in scenarios where only boundary or initial conditions are known, PINNs can infer the solution of a PDE across the entire domain by minimizing deviations from both the observed data and the underlying physics [11].

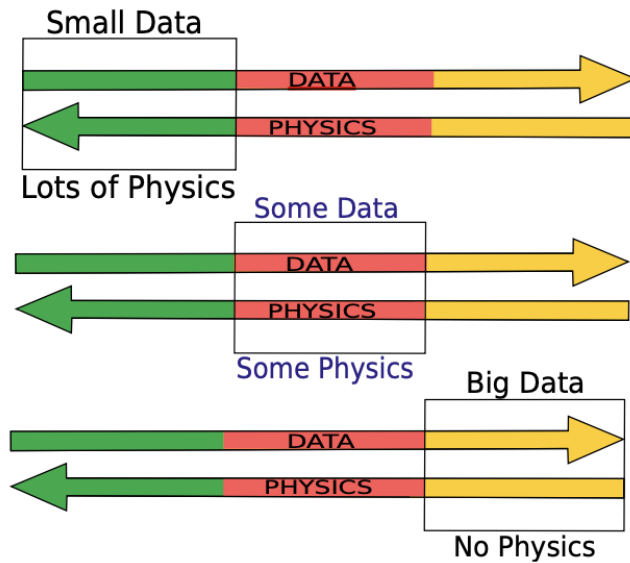


Figure 6 - Schematic to illustrate three possible categories of physical problems and associated available data. [9]

What sets PINNs apart is their reliance on **automatic differentiation**, a computational technique that allows for the exact calculation of derivatives with respect to both inputs and network parameters. This capability enables PINNs to compute PDE residuals directly, without the need for traditional discretization or numerical differentiation schemes, which are often prone to truncation errors and instability [2][7][11].

The versatility of PINNs extends beyond traditional PDEs to include stochastic equations, fractional-order systems, and problems with uncertain or missing parameters. They can solve forward problems - predicting system behavior from known inputs - as well as inverse problems, where the goal is to identify unknown parameters or even discover governing equations from observed data. With applications spanning fluid dynamics, material science, quantum mechanics, and beyond, PINNs represent a robust and scalable approach to solving some of the most challenging problems in modern science and engineering [7][11].

By integrating data and physical principles in a unified framework, PINNs not only address the limitations of traditional numerical methods but also open new frontiers for data-driven scientific discovery and modelings [11].

By leveraging their ability to integrate physics and data seamlessly, PINNs have found applications in fluid dynamics, quantum mechanics, reaction-diffusion systems, and beyond. This introduction sets the stage for a deeper exploration of their mathematical framework, computational techniques, and applications in subsequent sections.

1.3.1 Mathematical and Computational Foundations of PINNs

At the core of PINNs is their capability to approximate the solution of a physical system governed by Partial Differential Equations (PDEs). These PDEs are typically expressed in the general form:

$$F(u; \gamma) = 0$$

where u represents the unknown solution (e.g., temperature, velocity, or pressure), γ encapsulates the system parameters (e.g., diffusion coefficients, external forces), and F is a nonlinear operator that defines the governing equations of the physical system.

For instance, consider the **heat equation**, which models the diffusion of heat over time:

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

where α is the diffusion coefficient, ∇^2 is the Laplace operator, and $u(x, t)$ represents the temperature at spatial coordinate x and time t .

In PINNs, the solution $u(x, t)$ is approximated using a neural network $u_\theta(x, t)$, where θ denotes the trainable parameters of the network. The training objective is to minimize a composite loss function that balances multiple components:

$$\mathcal{L}(\theta) = \omega_F \mathcal{L}_F + \omega_B \mathcal{L}_B + \omega_d \mathcal{L}_{data}$$

Where:

- $\mathcal{L}_F = \|F(\mathbf{u}_\theta; \boldsymbol{\gamma})\|^2$: Enforces the residual of the PDE, ensuring that the neural network approximates a valid solution to the governing equations.
- \mathcal{L}_B : Penalizes deviations from boundary conditions.
- \mathcal{L}_{data} : Measures the discrepancy between the neural network predictions and any observational data available.

For example, in the Burgers' equation, a nonlinear PDE often used as a simplified model for fluid dynamics:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - v \frac{\partial^2 u}{\partial x^2} = 0$$

PINNs optimize the residual \mathcal{L}_F for the PDE, while simultaneously incorporating initial and boundary conditions into \mathcal{L}_B [2][11].

1.3.2 The role of automatic differentiation

A key computational innovation in PINNs is the use of **automatic differentiation (AD)** to compute derivatives with machine precision. AD evaluates derivatives of the neural network outputs (e.g.,) with respect to inputs directly within the backpropagation framework of deep learning. This avoids the numerical errors and instability often encountered in finite difference or finite element schemes, where derivatives are approximated via discretization. For example, in the Burgers' equation:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - v \frac{\partial^2 u}{\partial x^2} = 0$$

the terms $\frac{\partial u}{\partial t}$, $\frac{\partial u}{\partial x}$ and $\frac{\partial^2 u}{\partial x^2}$ are directly computed using AD applied to $u_\theta(x, t)$. This enables the precise evaluation of the residual \mathcal{L}_F , ensuring the neural network respects the governing physics at every training point in the domain [7][11].

1.3.3 Challenges and Advances in Training PINNs

Despite their promise, the training of PINNs presents significant challenges that stem from the integration of physical constraints into neural network architectures, the high dimensionality of many scientific problems, and the stiff nature of some PDEs. This section explores the computational and optimization hurdles faced in training PINNs and highlights recent advancements designed to overcome these obstacles.

Challenges in training PINNs

One of the primary challenges in training PINNs is their **high computational expense**. Unlike traditional machine learning models, the loss function in PINNs is composed of multiple terms, including residuals of the governing equations, penalties for boundary conditions, and errors with respect to observational data. For many scientific problems, especially those in high-dimensional spaces, evaluating these terms requires sampling a large number of collocation points within the domain. As the complexity of the system increases the computational cost grows significantly, often exceeding the limits of conventional hardware resources. Furthermore, PDEs with higher orders of differentiation exacerbate this issue by requiring additional computations of derivatives via automatic differentiation [7][11].

Another significant difficulty lies in **balancing the multiple terms within the composite loss function**. The contributions of physics residuals, boundary conditions, and observational data must be appropriately scaled to ensure effective training. Improper scaling can lead to optimization problems where the neural network overfits one component of the loss function, such as the observational data, while neglecting others like the physics residuals. This imbalance can result in solutions that appear to fit the data but violate the underlying physical laws, undermining the core purpose of the PINN frameworks [7][11].

Stiffness in PDEs, such as those describing fluid dynamics or chemical reactions, presents another challenge. Stiff PDEs are characterized by widely varying solution scales across space or time, making it difficult for gradient-based optimizers to converge efficiently. In such cases, the optimization process can suffer from vanishing gradients in regions where the solution is less sensitive, while other regions with steep gradients remain unresolved. This behavior complicates training and often necessitates domain-specific adaptations [11][12].

Overfitting to collocation points is an additional concern, particularly in scenarios where data is sparse or noisy. PINNs rely heavily on collocation points to enforce the physics-informed loss, but in cases where observational data is limited, the network can overfit to these points, failing to generalize across the entire domain. This limitation is especially problematic in inverse problems where the goal is to infer unknown parameters or discover governing equations [2][11].

Advances in Addressing Training Challenges

To address these challenges, several advancements in methodology and architecture have been developed. One of the most significant innovations is the use of adaptive loss weighting strategies. These approaches dynamically adjust the relative weights of the residual, boundary, and data loss terms during training to ensure a balanced optimization process. For example, methods based on the **Neural Tangent Kernel (NTK)** have been proposed to focus the optimizer's attention on regions of the domain where residuals are large, thereby improving convergence. The Neural Tangent Kernel (NTK) framework helps analyze and optimize neural network training dynamics. In the context of PINNs:

NTK-based methods dynamically reweight the loss terms associated with residuals from the differential equations, boundary conditions, and data points.

The optimizer gives more weight to regions with larger residuals, effectively prioritizing areas where the solution deviates the most from the governing equations or boundary conditions.

This ensures a more balanced and efficient training process, especially in cases where some regions of the domain have stiff gradients or are more challenging to model [14].

Other techniques, such as residual-based attention mechanisms, assign higher weights to regions with greater violations of physical constraints, ensuring that the network prioritizes these areas [11][12].

Advancements in domain decomposition techniques have also enhanced the scalability of PINNs. **Extended PINNs (XPINNs)** [Figure 7] partition the computational domain into smaller subdomains, each solved by an independent neural network. These subdomains are coupled through continuity conditions on state variables and PDE residuals at their interfaces, enabling parallel training and reducing the computational burden. This approach is particularly effective for multi-scale problems, where localized features require finer resolution. Similarly, Conservative PINNs (CPINNs) extend this methodology by explicitly enforcing conservation laws at subdomain boundaries, making them ideal for applications involving conservation-based PDEs like the Navier-Stokes equations [11][12][15].

Hybrid methods that combine PINNs with traditional numerical solvers offer another promising solution. In such frameworks, classical numerical methods, such as finite element or finite difference approaches, are used to resolve stiff or highly localized features, while PINNs model broader regions of the domain where data is sparse, or parameters are uncertain. This combination balances computational efficiency with the flexibility of PINNs,

making it particularly suitable for problems involving high stiffness or irregular domains [2][11].

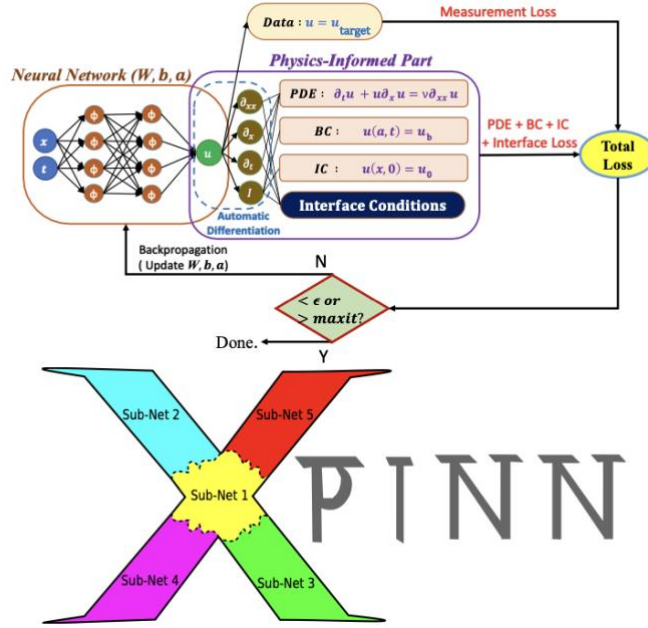


Figure 7 - The top figure is the schematic of XPINN sub-net employed in a subdomain where NNs and PI part for Burgers equation are shown. The bottom figure shows the irregularity shaped sub-domain division in 'X'-shaped domain.[15]

Sequential training and transfer learning have further addressed issues related to temporal dynamics and initialization. Sequential training divides the temporal domain into smaller intervals and trains the network sequentially over these intervals, preventing earlier time steps from dominating the optimization process. Transfer learning, in which a pre-trained PINN is fine-tuned for new problems or extended domains, has also shown promise in accelerating convergence and reducing computational costs [11].

Another effective strategy involves the use of surrogate models. Low-fidelity approximations of the solution can guide the initialization of the neural network, reducing the computational burden during the early stages of training. These surrogate models provide a coarse but computationally inexpensive representation of the system, which can later be refined by the PINN [11][12].

1.3.4 The Role of Parallelization and High-Performance Computing

The scalability of PINNs for large-scale or high-dimensional problems has been significantly improved by leveraging high-performance computing. Domain decomposition techniques, such as XPINNs, have benefited from **GPU parallelization** [Figure 8], where each subdomain is trained independently on separate processing units. Tensor-based architectures

and distributed training frameworks have further extended PINNs' applicability to higher-dimensional problems, such as those in \mathbb{R}^{100+} , opening doors to previously intractable scientific applications [11][12].

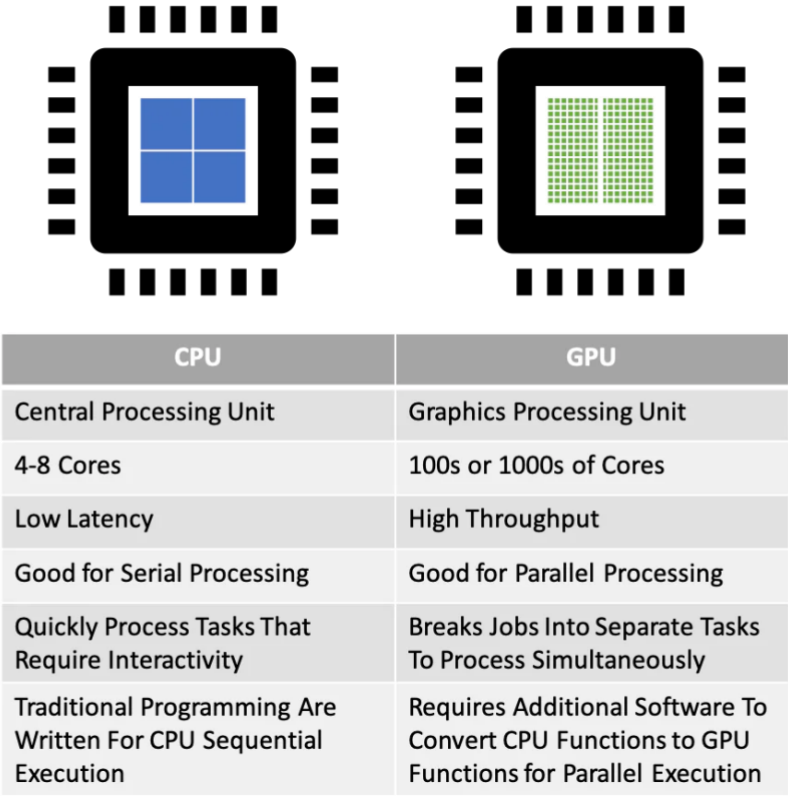


Figure 8 - Explanation of the difference between a CPU and a GPU [13]

2. Milling Process

The **milling process** [Figure 9] is a critical and widely utilized manufacturing method in industries ranging from automotive to aerospace. This paper provides an in-depth examination of the milling process, its principles, techniques, and applications. The discussion covers fundamental concepts, equipment, and parameters that influence the outcome of the milling operation. This comprehensive overview aims to enhance understanding and optimize the efficiency of milling operations.



Figure 9 - This image showcases the milling process. The application of a cutting fluid, visible as a spray, serves to cool the tool, reduce friction, and remove chips from the cutting area. [16]

Milling is a machining process in which a rotating cutting tool removes material from a workpiece by advancing in a direction at an angle with the axis of the tool [17]. Unlike turning processes, where the tool is stationary and the workpiece rotates, milling involves dynamic interaction between the tool and the workpiece, leading to versatility in creating complex geometries. The importance of milling lies in its ability to produce high-precision components efficiently and with repeatability [18].

2.1 Principles of Milling

At its core, milling relies on the cutting action of rotating multi-point tools. The cutting process is governed by several principles:

- **Material Removal:** Material is removed in the form of small chips by relative motion between the cutting tool and the workpiece.
- **Feed Rate:** The speed at which the workpiece is fed to the cutting tool affects surface finish and production time.

- **Depth of Cut:** The thickness of material removed in a single pass significantly influences cutting forces and tool wear.
- **Cutting Speed:** The peripheral speed of the cutting tool impacts efficiency and heat generation.

2.2 Types of Milling Operations

Milling operations are categorized based on the orientation of the cutting tool and the direction of feed relative to the workpiece, each designed to achieve specific machining objectives:

- **Face Milling:** In this operation, the cutting tool's axis is perpendicular to the surface being machined. It is primarily used for machining flat surfaces, providing a smooth finish and high material removal rates.
- **Peripheral Milling:** Here, the cutting tool's axis is parallel to the surface being machined. This operation is ideal for cutting slots, creating profiles, and machining edges.
- **End Milling:** A versatile operation that employs end mills to perform tasks such as contouring, slotting, and drilling. This technique is widely used for creating intricate features and complex shapes.
- **Angular Milling:** This operation involves cutting at an angle to the workpiece surface, making it suitable for producing features like chamfers, grooves, or tapered surfaces.
- **Form Milling:** This specialized operation uses specially shaped cutters to produce contours, profiles, or complex geometries directly on the workpiece.

2.3 Equipment and Tooling

The milling process utilizes a diverse array of equipment and tooling, each designed to optimize performance for specific applications:

- **Milling Machines:**
 - **Horizontal Milling Machines:** Equipped with horizontally oriented spindles, these are ideal for heavy-duty tasks such as cutting slots and gear teeth.
 - **Vertical Milling Machines:** Feature vertically oriented spindles, commonly used for face milling, drilling, and end milling.

- **Universal Milling Machines:** Capable of performing a variety of operations with adjustable spindle orientations and rotary tables.
- **Cutting Tools:**
 - **End Mills:** Versatile tools available in various geometries (flat, ball nose, corner radius) for contouring, slotting, and profiling.
 - **Face Mills:** Wide tools designed for high-efficiency face milling with superior surface finishes.
 - **Slot Drills:** Specialized end mills used for precision slot cutting.
 - **Roughing End Mills:** Designed with serrated edges for efficient material removal during roughing operations.
 - **T-Slot Cutters:** Used for machining T-shaped slots commonly found in machine tool tables.
 - **Form Cutters:** Custom-designed tools for producing specific profiles or shapes.
 - **Thread Mills:** Precision tools for cutting internal or external threads.
 - **Fly Cutters:** Single-point tools used for face milling large surfaces with high precision.
- **Tool Materials and Coatings:**
 - **High-Speed Steel (HSS):** Provides excellent toughness, commonly used for general-purpose applications.
 - **Carbide:** Offers superior wear resistance and cutting speed, ideal for high-performance machining.
 - **Ceramic and Cermet Tools:** Used for high-temperature applications and machining hard materials.
 - **Coatings:** Tools are often coated with materials such as titanium nitride (TiN) or diamond-like carbon (DLC) to enhance wear resistance and reduce friction.

2.4 Parameters Influencing Milling

Several parameters affect the efficiency and quality of milling:

- **Tool Geometry:** Tool material, coating, and design impact cutting efficiency.
- **Cutting Environment:** Use of cutting fluids enhances lubrication, cooling, and chip removal.
- **Workpiece Properties:** Hardness, ductility, and thermal conductivity of the workpiece material dictate cutting parameters.

- **Machine Stability:** Vibrations and machine rigidity influence dimensional accuracy and surface quality.

2.5 Tool Wear

Predicting tool wear in milling is pivotal because tight dimensional tolerances can be held and surface integrity preserved so it can be intervened before quality escapes the process window [19]; productivity can be increased and cost lowered by enabling proactive tool changes, stable parameter selection and fewer unplanned stoppages, as consistently reported in tool-condition monitoring studies [20]; reliability and safety are improved by preventing edge breakage and collateral damage to fixtures and high-value parts, with change points aligned to standardized tool-life criteria from ISO 8688.

Fundamentally, wear evolves from the interplay of:

- Workpiece material and microstructure: thermophysical properties and abrasive tendencies in Ti and Ni superalloys accelerate adhesion, diffusion, notch and flank wear [21].
- Tool substrate, coating and edge micro-geometry: hot chemical stability, low-friction coatings, and lead angles reshape contact length, friction and local stress fields, often extending life sustainability [22].
- Cutting parameters and engagement: cutting speed V remains the strongest accelerator of thermally driven wear, while feed per tooth thickness and radial depth set forces and sliding distance [23].
- Cooling and lubrication: dry, flood and cryogenic delivery change interfacial temperature/chemistry and can markedly reduce wear when applied appropriately [25][26].
- Operating history: length of cut, thermal cycling and the interrupted nature of milling promote micro-chipping and fatigue [23].

2.5.1 Taylor Equation

In order to estimate and plan around tool wear in milling, Taylor's tool-life equation was developed as a compact empirical law that links cutting speed to the time (or length of cut) until a defined end-of-life, providing a practical backbone for speed selection, planned tool changes, and baseline monitoring models [24]:

$$VT^n = C$$

V: cutting speed (m/min or SFM) set at constant feed per tooth, radial/axial engagement, tool geometry and cooling

T: tool life measured to a specific criterion (minutes of engaged cutting or equivalent length of cut to a flank-wear/chipping limit)

n: wear-sensitivity exponent that reflects mechanism dominance and how sharply life falls with speed

C: speed constant tied to the tool–workpiece–setup and the chosen life criterion

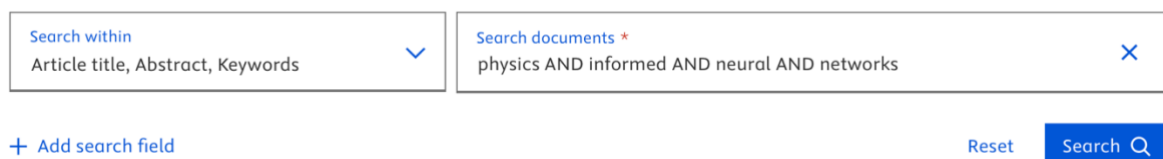
3. State of the Art

The research methodology employed in this thesis is outlined below. To establish a foundational understanding of the topics, essential definitions (e.g., Milling Process and Neural Networks) were obtained from credible, informative, and reputable online sources and from academic books used to several universities' course. The study followed a top-down approach, beginning with broad keywords and progressively narrowing the scope through more specific queries, utilizing the Scopus database as a primary tool.

Initially, the research aimed to provide a comprehensive overview of existing literature on Physics Informed Neural Networks (PINNs), Milling processes, Neural Networks, and Machine Learning. Keywords were searched within the "Article Title," "Abstract," and "Keywords" sections of research articles, and subsequently combined to refine and target the results more effectively.

3.1 Research Process

The following section describes the steps taken during the research process [Figure 10]:



The image shows a search interface with two main input fields. The first field, labeled 'Search within', has a dropdown menu with 'Article title, Abstract, Keywords' selected. The second field, labeled 'Search documents *', contains the query 'physics AND informed AND neural AND networks'. Below these fields are three buttons: '+ Add search field', 'Reset', and 'Search' (with a magnifying glass icon).

Figure 10 - PINNs reseach on Scopus

A search so general led to 4,762 documents being found [Figure 11] (and at the time of the research there were more than 2500 preprints on the topic). An increasing number of articles have been developed in the last 20 years, reflecting the increasing interest in the subject.

Talking about the application field, the main concerning areas are Engineering,

followed by Computer Science and Mathematics [Figure 12]. In addition, most of the documents found are from China, US and Germany [Figure 13].

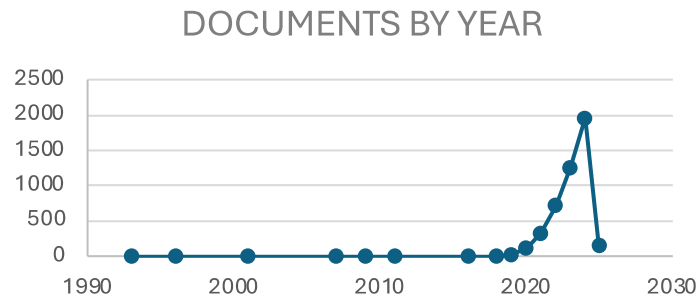


Figure 11 – PINNs Articles per year



Figure 12 - PINNs Articles by subject area

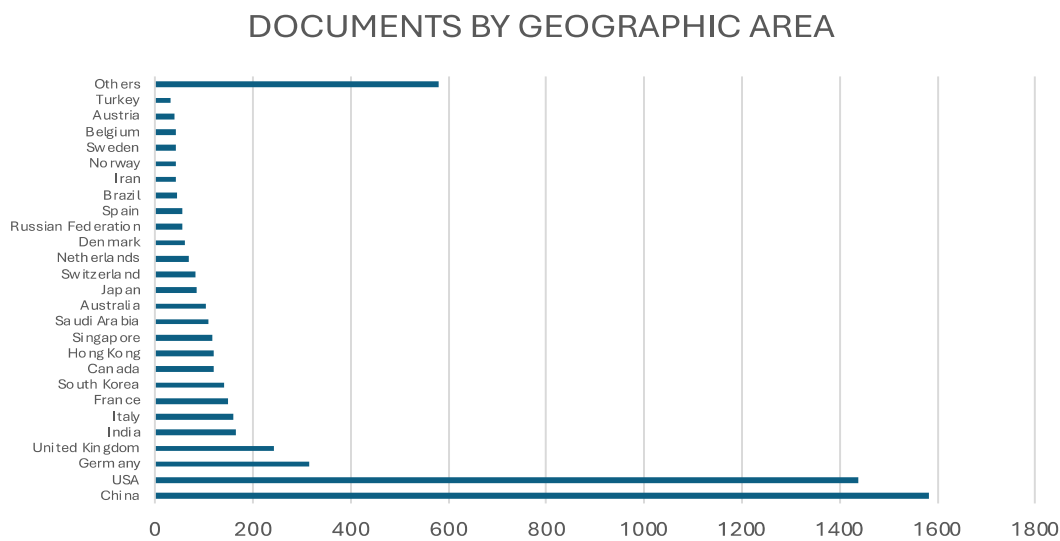


Figure 13 - PINNs articles by geographic area

I have then focused on searching sources for the PINNs applied at the milling process [Figure 14].

The screenshot shows a Scopus search interface. At the top, there are two search bars. The left bar is labeled 'Search within' and has a dropdown menu with 'Article title, Abstract, Keywords'. The right bar is labeled 'Search documents' and contains the query 'physics AND informed AND neural AND networks AND milling'. Below the search bars, there is a '+ Add search field' button, a 'Reset' button, and a 'Search' button with a magnifying glass icon.

Figure 14 - PINNs AND Milling Process research on Scopus

The search led to 5 documents being found and 1 preprint. The articles were written between the 2022 and 2023.

Three of them are from China, Two from US and One from India. The preprint is from China too.

3.2 Bibliographic research

First Query:

- Query “physics” AND “informed” AND “neural” AND “networks”
- 4840 documents found

The documents in Table 1 have been chosen as the most relevant, focusing on the development of PINNs and its use in manufacturing processes. These publications cover the most recent and popular ways to develop a PINNs and define which would be some important applications of it.

To identify the most relevant contributions, I prioritized publications with the highest number of citations, as these reflect the works most recognized and influential within the field. In addition, I considered papers published in leading journals and conferences, as well as those that introduced novel approaches or provided comprehensive reviews. This ensured that both the most authoritative and the most innovative studies were included in the analysis.

TITLE	AUTHORS	DESCRIPTION	REF
Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations.	M. Raissi, P. Perdikaris and G.E. Karniadakis	This paper explores the use of Physics-Informed Neural Networks (PINNs) to discover governing partial differential equations (PDEs) from data. By integrating physics-based constraints into deep learning models, the study demonstrates how PINNs can infer underlying mathematical structures from sparse and noisy measurements. Applications include the Burgers' and Navier-Stokes equations, where PINNs successfully reconstruct the equations	[2]

		despite limited data availability. The paper also discusses challenges such as noise sensitivity, data sparsity, and network architecture selection. The findings highlight the potential of PINNs for data-driven scientific discovery, particularly in fields where acquiring large datasets is impractical.	
Scientific Machine Learning through Physics-Informed Neural Networks	George Em Karniadakis	This review provides a comprehensive overview of Physics-Informed Neural Networks (PINNs), a deep learning framework that integrates scientific laws into neural networks for solving complex partial differential equations (PDEs). The paper examines different approaches for enhancing PINNs, including variations in activation functions, network structures, and optimization techniques. While PINNs have shown significant advantages in handling inverse and forward problems, challenges such as computational cost, hyperparameter sensitivity, and theoretical uncertainties remain. The study also explores advanced variants like CPINNs and hp-VPINNs, which aim to improve accuracy and efficiency. Overall, PINNs continue to be a promising tool for bridging machine learning with numerical analysis.	[3]
Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next.	Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi and Francesco Piccialli	This study delves into the evolution and customization of Physics-Informed Neural Networks (PINNs), comparing them with traditional numerical methods. The paper categorizes different PINN variants, including Physics-Constrained Neural Networks (PCNNs) and Conservative PINNs (CPINNs), and explores strategies to optimize their performance. Key challenges discussed include training stability, computational efficiency, and the integration of physical constraints with data-driven learning. Despite these limitations, PINNs offer a flexible and effective alternative for solving high-dimensional and complex physical problems. The paper underscores the need for further advancements in loss function design, optimization algorithms, and theoretical analysis to enhance their reliability.	[7]

Physics Informed machine Learning.	George Em Karniadakis, Lu Lu, Yannis Kevrekidis, Paris Perdikaris	This paper examines the integration of physics-based constraints into machine learning models, focusing on how Physics-Informed Machine Learning (PIML) can enhance model generalization and predictive accuracy. Traditional numerical methods like finite elements often struggle with high-dimensional data, noise, and ill-posed problems, whereas physics-informed approaches embed governing equations directly into neural networks. The study explores applications in geophysics, climate modeling, and fluid dynamics, demonstrating how physics-based learning can outperform purely data-driven models. Additionally, it discusses techniques such as kernel-based regression, Bayesian inference, and uncertainty quantification, which help refine predictions and improve interpretability in scientific computing.	[11]
Physics-Informed Neural Networks and Extensions.	Maziar Raissi, Paris Perdikaris, Nazanin Ahmadi, and George Em Karniadakis	This paper reviews recent advancements in Physics-Informed Neural Networks (PINNs), with a particular focus on data-driven discovery of differential equations. The authors explore improvements such as adaptive weighting strategies, which balance physics-based and data-driven losses, as well as domain decomposition techniques that enable parallelized computations. The study also highlights PINNs' potential in solving inverse problems, where unknown parameters or missing physics must be inferred from observations. Additional topics include stochastic PDEs, uncertainty quantification, and hybrid models that blend machine learning with traditional solvers. While challenges remain in training efficiency and multi-scale problem handling, PINNs continue to evolve as a powerful tool for scientific machine learning.	[12]
Neural Tangent Kernel: Convergence and Generalization in Neural Networks.	Arthur Jacot, Franck Gabriel, Clement Hongler	This study introduces the Neural Tangent Kernel (NTK), a theoretical framework that helps explain the training dynamics and generalization properties of deep neural networks. NTK theory suggests that, in the infinite-width limit, neural networks behave similarly to kernel	[14]

		methods, following predictable optimization paths. The authors examine how NTK can offer insights into function approximation, convergence, and stability, particularly in the context of Physics-Informed Neural Networks (PINNs). The findings highlight NTK's potential to guide the design of more efficient and theoretically grounded deep learning models, particularly for solving differential equations in scientific applications.	
Extended physics-informed neural networks (XPINNs): A generalized space-timedomain decomposition based deep learning framework for nonlinear partial differential equations.	Ameya D. Jagtap, George Em Karniadakis	This paper presents Extended PINNs (XPINNs), an advanced framework that enhances traditional Physics-Informed Neural Networks (PINNs) using domain decomposition techniques. By dividing the computational domain into smaller subdomains, each handled by separate neural networks, XPINNs allow for parallel computation, improved scalability, and more efficient handling of complex geometries. The study highlights how XPINNs simplify interface conditions, improve numerical stability, and optimize neural network architecture per subdomain. Applications include multi-scale physics problems, discontinuous solutions, and high-dimensional PDEs, where XPINNs offer superior performance compared to standard PINNs.	[15]

Table 1 - Research papers analyzed for Query 1

The papers focus on the use of physics-informed neural networks (PINNs) for solving nonlinear partial differential equations and enhancing scientific machine learning. Various approaches integrate deep learning with physical constraints, leveraging automatic differentiation to enforce PDE solutions. Some studies apply PINNs to fields such as fluid dynamics, quantum mechanics, and reaction-diffusion systems, while others explore extended variants like XPINNs and CPINNs to improve scalability and parallelization in complex domains. Further research focuses on uncertainty quantification and Bayesian PINNs to enhance robustness in inverse problems. Additionally, some works investigate the Neural Tangent Kernel (NTK) framework to analyze PINNs' convergence and generalization properties.

Second Query:

- Query “physics” AND “neural” AND “networks” AND “milling”
- 21 documents found

Eight papers were selected for inclusion in this study. Of these, five publications focus on the application of Physics-Informed Neural Networks (PINNs) for tool wear prediction, providing insights into their effectiveness in modeling and forecasting wear behavior under various machining conditions. Two papers explore the use of PINNs in estimating surface roughness during the milling process, highlighting the potential of these neural networks in characterizing and predicting surface quality based on machining parameters. The remaining paper is a study on how to detect unstable vibration during the milling process. Table 2 presents the titles and main contributions of the selected publications, offering a comprehensive overview of the relevant literature that underpins this study.

TITLE	AUTHORS	DESCRIPTION	REF
Physics-Informed Meta Learning for Machining Tool Wear Prediction	Yilin Li, Jinjiang Wang, Zuguang Huang, Robert X. Gao	This paper presents a physics-informed meta-learning (PIML) framework for predicting tool wear in machining operations. It integrates empirical equations into data-driven models to enhance interpretability and accuracy. The proposed model adapts to varying tool wear rates by employing a physics-informed loss term, ensuring optimization consistency. An experimental study on a milling machine validates the framework's effectiveness. The results show improved prediction accuracy compared to conventional machine learning approaches, demonstrating the potential of integrating physical principles with deep learning for smart manufacturing applications.	[31]
Hybrid Data-Driven and Model-Informed Online Tool Wear Detection in Milling Machines	Qian Yang, Krishna R. Pattipati, Utsav Awasthi, George M. Bolas	This study introduces a hybrid anomaly detection model that combines physics-based and machine learning models to monitor tool wear in real-time. The model integrates a decision tree or neural network with Page's cumulative sum test to detect tool degradation using numerical control machine measurements. The approach improves anomaly detection accuracy by fusing machine sensor data with domain knowledge. Experiments on milling	[32]

		machines validate the model, showing a 92% accuracy rate and reduced false alarms compared to standalone physics-based or data-driven methods.	
A Physics-Informed Machine Learning Model for Surface Roughness Prediction in Milling Operations	Pengcheng Wu, Haicong Dai, Yufeng Li, Yan He, Rui Zhong, Jinsen He	This paper proposes a physics-informed neural network (PINN) for predicting surface roughness in milling. By incorporating physical constraints into deep learning models, the approach reduces dependency on large datasets while improving accuracy. Experimental validation demonstrates that the proposed model achieves better performance compared to traditional machine learning methods. The study highlights the benefits of integrating physics-based knowledge with AI to enhance manufacturing quality and efficiency.	[33]
Milling Surface Roughness Prediction Based on Physics-Informed Machine Learning	Shi Zeng, Dechang Pi	This study presents a physics-informed deep learning (PIDL) method for predicting milling surface roughness. By integrating physics-based constraints into a CNN-GRU model, the approach enhances prediction accuracy while maintaining physical consistency. The method uses data augmentation through mechanistic modeling and employs a physics-guided loss function to enforce theoretical constraints. Experimental validation on open-source datasets shows improved accuracy over traditional machine learning models, making it a promising approach for smart manufacturing applications.	[34]
Physics-Informed Neural Networks With Weighted Losses by Uncertainty Evaluation for Accurate and Stable Prediction of Manufacturing Systems	Jiaqi Hua, Yingguang Li, Member, IEEE, Changqing Liu, Peng Wan, and Xu Liu	The paper presents a physics-informed neural network with weighted losses (PINN-WL) to improve the accuracy and stability of predictions in manufacturing systems. By dynamically adjusting model weights based on uncertainty evaluation, it balances the contributions of data-driven and physics-based models. The approach addresses challenges posed by noisy data and inaccurate physics models. Tested on tool wear prediction datasets, it outperforms existing methods in both accuracy and stability. The method has potential applications in tool life	[29]

		monitoring, part quality assurance, and production automation, contributing to more reliable decision-making in intelligent manufacturing.	
A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests	Dazhong Wu, Connor Jennings, Janis Terpenny, Robert X. Gao, Soundar Kumara	The paper compares machine learning algorithms for tool wear prediction in smart manufacturing. It evaluates random forests (RFs), support vector regression (SVR), and feed-forward backpropagation artificial neural networks (FFBP-ANNs) using data from 315 milling tests. The study finds that RFs outperform SVR and FFBP-ANNs in prediction accuracy, though with longer training times. The results highlight the potential of RFs for improving predictive maintenance in manufacturing by providing more reliable tool wear estimates. The research supports data-driven approaches for transitioning traditional manufacturing systems into intelligent, self-monitoring operations.	[36]
On-line chatter detection in milling with hybrid machine learning and physics-based model	M. Hossein Rahimi, Hoai Nam Huynh, Yusuf Altintas	The paper presents a hybrid approach for online chatter detection in milling, combining machine learning and physics-based models. It processes vibration data using short-time Fourier transforms and a convolutional neural network to classify machining states. A parallel physics-based method isolates chatter from forced vibrations using a Kalman filter. The integrated approach achieves a 98.9% success rate in chatter detection, reducing false alarms while improving accuracy. The system enables real-time adjustments to spindle speed, enhancing machining stability. The method is validated through extensive experiments, demonstrating robustness across different cutting conditions and tools.	[35]
Data-Driven Prognostics Using Random Forests: Prediction of Tool Wear	Dazhong Wu, Connor Jennings, Janis Terpenny,	The paper explores the use of random forests (RFs) for tool wear prediction in milling operations, comparing its performance with artificial neural networks (ANNs) and support vector regression (SVR). It highlights the advantages of RFs, such as handling multiple input variables and reducing	[37]

	Robert Gao ² , Soundar Kumara	overfitting. Using data from force, vibration, and acoustic emission sensors, the study finds that RFs provide more accurate predictions than ANNs and SVR, though with longer training times. The results suggest RFs as a promising approach for predictive maintenance, improving the reliability of manufacturing processes by anticipating tool wear more effectively.	
A physics-informed neural network integrating physical mechanisms and experimental data for predicting VOCs emission rates in machining workshops	Mengdan Qiao, Yang Yang. Ningbin Zhu, Yi Wang; Sikang Li, Qingfeng Cao, Yanghui Hou, Bo Qian	This paper introduces a physics-informed neural network (VERF-PINN) to predict volatile organic compound (VOC) emission rates from metalworking fluids in machining workshops, combining an improved Langmuir–Maxwell evaporation model with data-driven learning for accuracy and interpretability. The authors first run single-factor and orthogonal experiments across process and environmental variables to build a tailored dataset and identify key drivers (notably workpiece temperature, ambient temperature, MWF type, and speed), then embed a semi-empirical emission formula both as an input feature and as a physics-informed loss.	[38]
An intelligent tool wear monitoring model based on knowledge-data-driven physics-informed neural network for digital twin milling	Xiaohui Fang; Qinghua Song*; Xiaojuan Wang; Zhenyang Li; Haifeng Ma; Zhanqiang Liu.	The paper builds a tool-wear predictor that mixes real machining data with simple physics rules, so predictions stay realistic even with limited data. It learns from common sensors (forces, vibration, acoustic emission), but also enforces that wear can't decrease and must follow a reasonable wear-rate law. Tested on several public milling datasets, it beat comparable neural nets without physics by about 10–15% error reduction. It runs fast enough for real-time use in a milling digital twin (milliseconds inference), helping schedule maintenance and avoid failures.	[39]

Table 2 - Research papers analyzed for Query 2

The collection of research papers found in Query 2 has been examined to identify the physical laws and datasets utilized in their investigations. The objective was to understand the theoretical frameworks supporting various machine learning and physics-informed

models applied to manufacturing processes, particularly in tool wear prediction, surface roughness estimation, and chatter detection. The reviewed studies employ a combination of experimental datasets, such as the **PHM 2010 Dataset**, **NASA Milling Dataset [28]**, and **Monte-Carlo Simulations**, alongside well-established physical laws, including **Taylor's tool wear life equation**, **cutting force models**, and **differential equations governing dynamic stability in milling operations**. Notably, some papers adopt purely data-driven approaches, while others integrate physics-based modeling to enhance prediction accuracy. The specific details regarding the datasets and physical laws used in each study are systematically presented in Table 3.

TITLE	DATASETS	PHYSIC LAWS
Physics-Informed Meta Learning for Machining Tool Wear Prediction	<ul style="list-style-type: none"> PHM 2010 Dataset 	<p>Taylor's tool wear life equation. Simplified as:</p> $\frac{d\theta}{dt} = C N^m$ <p>C: Empirical wear coefficient m: wear rate exponent N: force dθ/dt: tool wear rate</p>
Hybrid Data-Driven and Model-Informed Online Tool Wear Detection in Milling Machines	<ul style="list-style-type: none"> Monte-Carlo simulation Experimental Data 	$F_t = K_t A + \mu H W A_t$ <p>F_t: Tangential cutting force (N) K_t: Tangential force coefficient (N/mm²), which accounts for cutting conditions A: Chip area (mm²), representing the cross-sectional area of the chip being removed μ: Friction coefficient, which quantifies the resistance due to tool-workpiece contact H: Material hardness (HRC), which affects tool wear and force generation W: Tool wear (mm), representing the extent of tool degradation ap : Depth of cut (mm), which influences the material removal rate and cutting forces.</p>
A Physics-Informed Machine Learning Model for Surface Roughness Prediction in Milling Operations	<ul style="list-style-type: none"> Experimental Data 	<p>In this paper it is used the equation for surface roughness prediction in the milling process</p> $R_a = C R_z = C(R_0 + \Delta h)$

		<p>C: Proportional Coefficient</p> <p>A: constant that relates the maximum roughness height R_z to the arithmetic roughness R_a .</p> <p>This coefficient is often derived from experimental calibration.</p> <p>R_z: Maximum Roughness Height (μm)</p> <p>The difference between the highest peak and lowest valley over a given sampling length of the machined surface.</p> <p>R₀: Maximum Residual Height of the Surface (μm).</p> <p>The theoretical roughness height left after the milling process due to tool movement and cutting geometry.</p> <p>Δh: Additional Surface Deformation (μm).</p> <p>Represents the effect of plastic deformation and elastic recovery of the material.</p>
Milling Surface Roughness Prediction Based on Physics-Informed Machine Learning	<ul style="list-style-type: none"> • S45C Steel Milling Dataset 	<p>In this paper, as well as in the previous one, it is used the equation for surface roughness prediction in the milling process.</p>
Physics-Informed Neural Networks With Weighted Losses by Uncertainty Evaluation for Accurate and Stable Prediction of Manufacturing Systems	<ul style="list-style-type: none"> • Ideahouse Dataset • NASA Milling Dataset 	<p>Among the others the tool wear equation used in this paper is the following, developed by Z. Palmi [27]:</p> $\frac{dm}{dL} = \rho \frac{v_c}{dV/dt} = C_1 + C_2 e^{\left(-\frac{Q}{R\theta(W)}\right)}$ <p>dm/dL: This term represents the rate of mass loss due to tool wear as a function of the cutting length.</p>

		<p>ρ: The mass density of the tool material, which affects the total mass loss due to wear.</p> <p>v_c: The velocity at which the cutting tool moves relative to the workpiece.</p> <p>dV/dt: The rate at which the volume of the tool is lost due to wear.</p> <p>C_1, C_2: These coefficients depend on the hardness of the tool material and the normal stress developing on the tool's flank.</p> <p>Q: The energy barrier that must be overcome for wear processes such as diffusion or oxidation to occur.</p> <p>R: Universal Gas Constant</p> <p>$\theta(W)$: The temperature at the tool flank, which increases as wear progresses.</p>
A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests	<ul style="list-style-type: none"> PHM 2010 Dataset 	In this paper the Tool Wear prediction is only data driven.
On-line chatter detection in milling with hybrid machine learning and physics-based model	<ul style="list-style-type: none"> Experimental Data 	<p>In order to analyse the chatter in milling process, for this paper has been used the following differential equation:</p> $m \frac{d^2 x}{dt^2} + c \frac{dx}{dt} + kx = F(t)$ <p>M: is the mass of the system, c: damping coefficient, k: stiffness, x: displacement, $F(t)$: external force.</p>

Data-Driven Prognostics Using Random Forests: Prediction of Tool Wear	<ul style="list-style-type: none"> • Li et al. (2009) 	In this paper the Tool Wear prediction is only data driven.
--	--	---

Table 3 - Datasets & Physical Laws for Query 2 papers

4. PINNs in Milling Process

This chapter describes the case study employed to evaluate the effectiveness of a physics-informed neural network (PINN) for predicting tool wear during a milling process [29]. The experimental methodology and the resulting dataset are first outlined. Subsequently, the governing physics equations integrated into the model are presented. Finally, the Python implementation of the PINN algorithm is detailed, and its predictive performance is analyzed.

4.1 Experimental method and dataset

4.1.1 Milling machine and setup

Experiments were executed on a Matsuura MC-510V [Figure 15] operating at 200 m min^{-1} cutting speed.



Figure 15 - Matsuura MC-510V [16]

A 70 mm face-mill with six multilayer-coated KC710 carbide inserts was fitted for every run. Sixteen cutting conditions were tested:

- two depths of cut (0.75 mm, 1.5 mm),
- two feed rates (0.25 mm rev^{-1} , 0.5 mm rev^{-1})
- two work-piece materials (cast iron, stainless steel)

Each with a fresh insert set to follow wear from a sharp edge to its limit. After selected passes the machine was stopped, the cutter removed, and flank-wear land width VB was measured under a microscope [Figure 16]; VB was defined as the distance from the cutting edge to the end of the abrasive flank region. Repeating this pause-and-inspect sequence generated a

time-stamped series of VB values that documents the progressive degradation of each insert under well-controlled yet industrially relevant conditions, providing the ground-truth wear labels for subsequent modelling. [28]

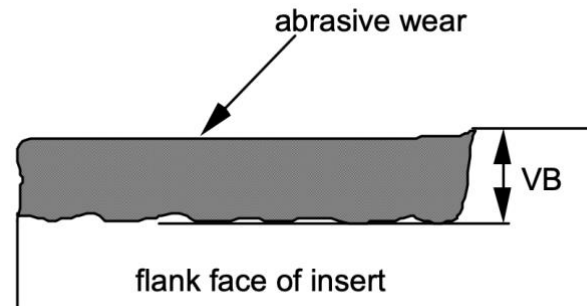


Figure 16 - Tool wear VB [28]

4.1.2 The Cutting Process

The cutting tool's engagement with the workpiece triggers a complex chain of physical phenomena, many of which can be monitored with sensors. As material in the primary shear zone plastically deforms and a chip forms, energy is released in the form of heat, cutting forces, vibration, and acoustic emission [Figure 17 - shear zones at tool/workpiece interface

- **Acoustic emission (AE)** consists of high-frequency stress waves (≈ 50 kHz to several MHz) generated when the metal's microstructure rearranges under deformation. AE originates in the primary and secondary shear zones where bulk deformation and chip-tool sliding occur and at the tool flank/workpiece interface due to friction. Because AE attenuates rapidly with distance, the sensor should be mounted as close to the cutting zone as protection allows. [28]
- **Vibration** is a lower-frequency oscillation (0–40 kHz) caused by dynamic variations in cutting forces stemming from periodic changes in tool geometry, chip formation, and built-up edge formation. Although the rigid Matsuura machining centre used here suppresses vibration compared with a lighter Bridgeport mill, these oscillations still influence tool wear and must be measured with a sensor positioned near the cut.
- **Cutting forces** arise in both shear zones and from friction at the chip-tool and workpiece-tool interfaces. They can be captured directly with force sensors beneath the workpiece, but this approach is often cumbersome and expensive. Instead, indirect measurement is preferred: monitoring spindle-motor current, feed-motor current, or overall power consumption. Because spindle current is proportional to torque, and torque to cutting force, this signal provides a convenient, low-cost proxy.

Feed-motor current supplies complementary information on feed forces, while power consumption offers an additional, closely related metric. Together, AE, vibration, and indirect force sensing provide a multifaceted picture of the cutting process and its influence on tool wear.

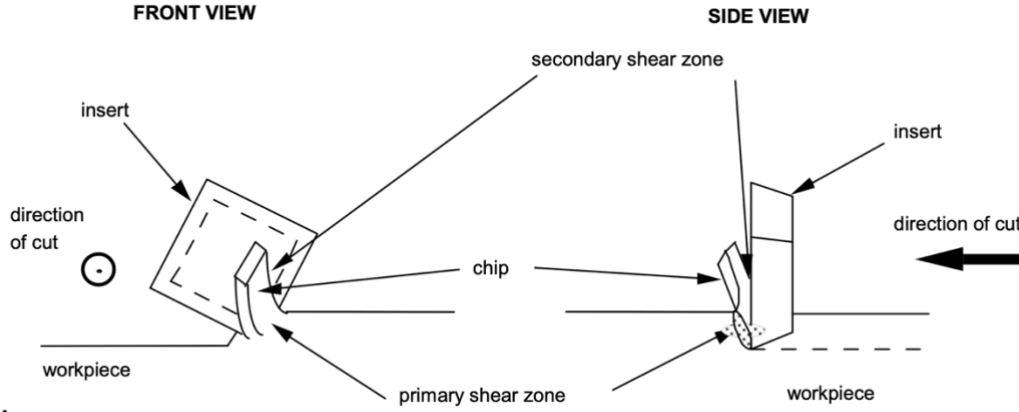


Figure 17 - shear zones at tool/workpiece interface [28]

4.1.3 Dataset Composition

The dataset couples synchronized process-signal streams with time-stamped flank-wear measurements gathered in a controlled milling campaign. A full $2 \times 2 \times 2$ factorial design combined two depths of cut (0.75 mm and 1.50 mm), two feed rates (0.25 mm rev^{-1} and 0.50 mm rev^{-1}) and two work-piece materials (cast iron and stainless steel), yielding sixteen distinct operating cases. Each case began with a fresh set of six multilayer-coated carbide inserts and continued until the tool approached its end-of-life wear limit. During cutting, five analogue channels: acoustic-emission energy and broadband vibration sensed at both the spindle and the machine table, plus spindle-motor current, were continuously conditioned into 8 ms root-mean-square envelopes and logged at 250 Hz, while raw high-bandwidth AE and vibration waveforms were archived for optional post-processing. At scheduled intervals the process was paused, the cutter removed, and the flank-wear land width V_b of each insert was measured under a microscope; these measurements were time-stamped and later aligned with the signal timeline by interpolation. Consequently, every trial contributes a multivariate time-series matrix ($\approx 250 \text{ samples s}^{-1}$ per channel) paired with a sequence of (t, V_b) labels and metadata describing the cutting parameters. Across all trials the collection totals roughly four hours of envelope data and more than one hundred individual wear measurements, providing a robust basis for modelling tool-wear progression under well-controlled yet industrially relevant conditions. [28]

Case	Depth of Cut [mm]	Feed [mm/min]	Material
1	1.5	0.5	1 – cast iron
2	0.75	0.5	1 – cast iron
3	0.75	0.25	1 – cast iron
4	1.5	0.25	1 – cast iron
5	1.5	0.5	2 – steel
6	1.5	0.25	2 – steel
7	0.75	0.25	2 – steel
8	0.75	0.5	2 – steel
9	1.5	0.5	1 – cast iron
10	1.5	0.25	1 – cast iron
11	0.75	0.25	1 – cast iron
12	0.75	0.5	1 – cast iron
13	0.75	0.25	2 – steel
14	0.75	0.5	2 – steel
15	1.5	0.25	2 – steel
16	1.5	0.5	2 – steel

Table 4 - Working conditions of NASA dataset with the relative specifications

4.1.4 Tool Wear

Maintaining a consistently sharp cutting edge is essential for achieving the tight dimensional tolerances and low surface roughness that define high-quality machined components. As the cutting edge dulls, it ploughs and smears the surface to a greater depth, promotes micro-tearing, and ultimately diminishes fatigue strength. The additional rubbing between a worn tool and the workpiece elevates the frictional heat at the cutting zone; temperatures can rise high enough to trigger undesirable metallurgical transformations. Tool degradation manifests in several forms: progressive edge rounding; crater wear on the rake face, generated by abrasive chip flow and diffusion; and flank wear on the clearance face, produced by sliding contact with the freshly machined surface. Among the cutting

parameters, spindle speed governs the wear rate most strongly, while feed and depth of cut also play influential roles through their impact on contact time, specific energy, and chip load.

In the present study tool condition is quantified by the flank-wear land width V_b , a metric widely adopted in ISO 3685 for tool-life testing [41]. V_b is defined as the perpendicular distance from the cutting edge to the outer boundary of the abrasive scar on the flank face. At predetermined intervals the machining operation was halted, the inserts were removed, cleaned, and examined under a calibrated optical microscope fitted with a graticule; the resulting V_b measurements were time-stamped and logged. Repeating this pause-and-inspect protocol throughout each cutting trial yielded a detailed chronology of wear growth that serves both as ground truth for data-driven models and as a benchmark for physics-based wear-rate predictions. A representative progression curve derived from one experimental case is shown in [Figure 18], illustrating the typical three-stage wear behavior: rapid break-in, steady-state linear growth, and an accelerated phase approaching tool failure[42].

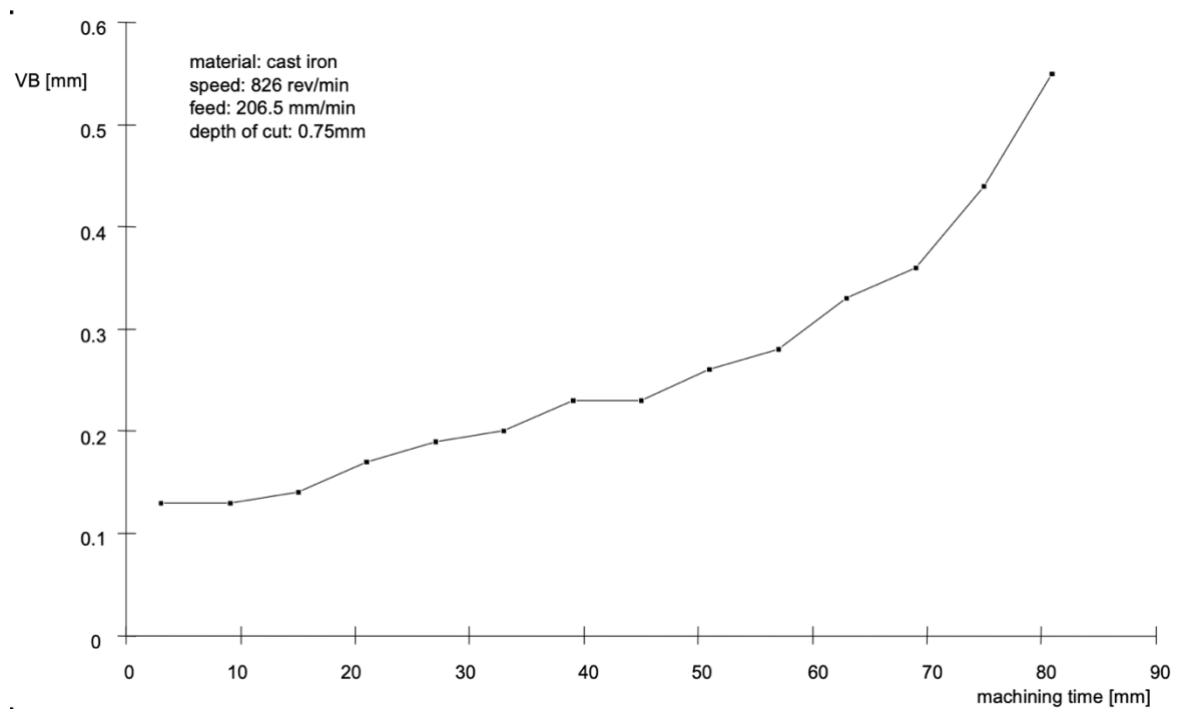


Figure 18 - Example of Tool Wear over time i one of the sixteen cases of the dataset [28]

4.2 Physics applied

In the proposed algorithm, the underlying physics of tool wear is embedded through the extended Taylor tool-life formulation, which relates the characteristic tool life T_f to the cutting parameters. Specifically:

$$T_f = KV^{-n}f_z^{-m}a_p^{-p}$$

where each parameter has a distinct physical meaning.

K is a scaling constant that defines the baseline magnitude of the tool life; it depends on the tool–workpiece material combination, coating, geometry, lubrication, and cutting environment [43].

V is the cutting speed [m/min] the primary parameter influencing thermal load, where higher values accelerate diffusion and oxidation wear [24][47].

n is the cutting speed exponent, which quantifies the sensitivity of tool life to variations in cutting speed [24].

f_z is the feed per tooth [mm/tooth], representing the chip thickness removed by each cutting edge during one revolution.

m is the feed exponent, which measures the effect of feed per tooth on tool wear through increased forces, stresses, and sliding distance [45].

a_p is the axial depth of cut [mm], indicating the thickness of the material layer removed along the tool axis.

p is the depth exponent, which captures how the axial engagement of the tool modifies heat flux, stresses, and wear rate [45].

This physical relationship is not imposed as a strict constraint but is incorporated into the learning process as a soft anchor through additional loss terms in the Physics-Informed Neural Network (PINN). The PINN itself is a fully connected neural network that receives as input the scaled time $\tau = t/T_f$ together with normalized process parameters (feed, depth of cut, material). The network predicts flank wear VB at any give τ , while three types of loss functions guide training:

1. A data loss that enforces agreement with measured wear values
2. An initial condition loss that penalizes non-zero predictions at the start of a test
3. A physics-based anchor that encourages the predicted wear at $\tau = 1$ to match a reference percentile of observed wear

In this way, the model combines empirical data with physics-inspired constraints, ensuring that predictions remain consistent with both experimental measurements and the known wear behaviour described by Taylor’s extended equation [24][46].

4.3 Python Implementation

4.3.1 Libraries

In order to implement the proposed framework, several Python libraries were employed to handle data processing, model construction, training, and visualization. Each library plays a specific role within the workflow, ranging from basic mathematical operations and file management to advanced neural network training and scientific plotting. The following list summarizes the libraries adopted, outlining their general purpose and the specific function they served in this work.

- **math**: used for elementary mathematical utilities and floating-point checks. In the code it was used for `math.isnan(...)` to verify the presence of valid numerical values when computing per-case maxima of tool wear
- **argparse**: used for robust command-line argument parsing and validation. In the code it was used for defining options such as input CSV path, number of epochs, output folders, and inner-loop settings (e.g., lambda search), enabling reproducible and scriptable experiments
- **os**: used for operating-system interactions (paths, directories, environment). In the code it was used for directory creation via `os.makedirs(...)` before saving figures and for constructing output file paths
- **time**: used for time handling and formatted timestamps. In the code it was used to produce human-readable log prefixes with the helper `now()` during training and evaluation
- **pathlib.Path**: used for object-oriented and cross-platform filesystem path management. In the code it was used to expand user paths and verify the existence of the dataset file prior to loading
- **dataclasses.dataclass**: used for concise, type-annotated data containers. In the code it was used to define *CaseData*, which encapsulates per-case arrays and metadata (times, VB labels, feed, depth of cut, material, etc.) for clean dataset handling

- **numpy**: used for high-performance array operations and numerical routines. In the code it was used for percentile computation of the reference wear value, random sampling in the lambda search, vectorized statistics (means/standard deviations), and error metrics
- **pandas**: used for tabular data ingestion, cleaning, grouping, and export. In the code it was used to read the CSV dataset, coerce column types, sort and group by case, assemble prediction tables for the hold-out sets, and write per-case metrics to Excel
- **torch**: used for tensor computation and automatic differentiation. In the code it was used to hold data as tensors on the selected device, to manage the optimization loop with *torch.optim.Adam*, to control randomness via *torch.manual*, and to clamp/transform values during the physics computations
- **torch.nn**: used for neural-network building blocks. In the code it was used to define the PINN architecture (*nn.Linear*, *nn.Tanh*) and the learnable Taylor parameters as *nn.Module* subclasses with registered *_nn.Parameter_s*
- **torch.nn.functional**: used for functional layers and loss/activation utilities. In the code it was used for the mean-squared-error loss *F.mse_loss* and for enforcing positivity of the Taylor scale factor with *F.softplus*
- **matplotlib.pyplot**: used for scientific plotting and result visualization. In the code it was used to generate per-case figures showing predicted vs. measured flank wear and absolute errors, and to save these plots as PNG files for reporting.

4.3.2 Tool Wear Prediction

The implementation presented predicts flank wear VB in end-milling by combining a compact neural regressor with a Taylor time scaling. The model ingests a **scaled time** τ and normalized process settings (feed, depth of cut (DOC) and a material identifier) and outputs a non-negative estimate of wear [Figure 19]. It has been decided to use the scaled time because under different speeds, feeds and DOCs, flank wear grows at very different rates and scaling by the **characteristic time** T_f (from a Taylor-type law) removes most of this rate effect, so that cases with similar physics collapse onto trajectories that are much more alike in shape.

```
# Compute Taylor time scale and scaled time  $\tau = t / T_f$ 
Tf = taylor_Tf(V, feed, doc, n, m, p, K)
tau = t / Tf
```

Figure 19 – T_f found as result of the *taylor_Tf* function and the calculation of the scaled time τ

In the code, the regressor f_θ is a fully connected network with tanh activations, and the output is clamped to zero from below to maintain physical plausibility:

$$\widehat{VB}(t) = f_{\theta}(\tau, \hat{f}, \hat{d}, m) \geq 0$$

It has been used a **tanh** activator because it is smooth, keeps neuron outputs between -1 and 1 , and is centered around 0 , which matches the normalized inputs and makes learning stable. The **activator** (activation function) is a rule applied to each neuron's output that adds the “bend” so the network can learn curves instead of just straight lines. [Figure 20]

```
class PINN_VB(nn.Module):
    """
    Simple fully-connected NN: input = [\tau, normalized feed, normalized DOC, material].
    Output = predicted VB at that \tau and settings.
    """
    def __init__(s, in_dim=4, width=64, depth=4):
        super().__init__()
        layers=[]; dims=[in_dim]+[width]*depth+[1]
        for i in range(len(dims)-2):
            layers += [nn.Linear(dims[i],dims[i+1]), nn.Tanh()]
        layers += [nn.Linear(dims[-2],dims[-1])]
        s.net = nn.Sequential(*layers)
    def forward(s,x): return s.net(x)
```

Figure 20 – Fully connected regressor for Flank-Wear prediction

4.3.3 Taylor's parameters prediction

The four **Taylor parameters** n , m , p , and K are learned jointly with the network weights. To keep them physically meaningful, the code uses smooth reparameterizations. A **sigmoid** maps n , m , and p into bounded intervals by squeezing raw values into a chosen range so the learned exponents stay physically meaningful. A **softplus** makes K strictly positive without hard clipping, which preserves physical validity and stabilizes optimization.

$$\begin{aligned} n &= \sigma(a_n)(n_{max} - n_{min}) + n_{min} \\ m &= \sigma(a_m)(m_{max} - m_{min}) + m_{min} \\ p &= \sigma(a_p)(p_{max} - p_{min}) + p_{min} \\ K &= \log(1 + e^{b_k}) \end{aligned}$$

a_n, a_m, a_p, b_k are unconstrained learnable variables. This construction allows standard gradient-based optimization while preventing nonphysical exponents or negative scales. Conceptually, the Taylor parameters govern how aggressively operating conditions accelerate or decelerate the “wear clock” trough T_f , and learning them lets the data determine the appropriate scaling. [Figure 21]

```

class TaylorParams(nn.Module):
    """
    Learnable Taylor-like parameters:
    n: exponent for cutting speed V
    m: exponent for feed
    p: exponent for DOC
    K: scale factor (positive via softplus)
    They are constrained to given ranges using a sigmoid mapping.
    """
    def __init__(s, use_extended=True):
        super().__init__(); s.use_extended = use_extended
        s.a_n = nn.Parameter(torch.tensor(0.0)) # unconstrained; will be squashed to N_RANGE
        s.a_m = nn.Parameter(torch.tensor(-2.0)) # idem for M_RANGE
        s.a_p = nn.Parameter(torch.tensor(-2.0)) # idem for P_RANGE
        s.b_K = nn.Parameter(torch.tensor(6.0)) # K made positive via softplus
    def forward(s):
        n = sigmoid_range(s.a_n, *N_RANGE)
        if s.use_extended:
            m = sigmoid_range(s.a_m, *M_RANGE)
            p = sigmoid_range(s.a_p, *P_RANGE)
        else:
            m = torch.zeros_like(n); p = torch.zeros_like(n)
        K = F.softplus(s.b_K)
        return n, m, p, K

```

Figure 21 – Learnable variables a_n, a_m, a_p, b_k and application of the sigmoid/softplus to them

4.3.4 Loss Function

As implemented, training minimizes a single objective that combines data fit, an initial-condition penalty, and a physics anchor:

$$L = L_{data} + \lambda_{ic}L_{ic} + \lambda_T L_{anchor}$$

This objective is evaluated on mini-batches and minimized with the Adam optimizer, which adaptively sets a step size for each parameter by keeping running averages of the gradient and the squared gradient (second moment); this usually gives fast, stable learning with little tuning beyond a learning rate and two coefficients.

Both the network weights and the Taylor parameters are updated. The weights λ_{ic} (initial conditions) and λ_T (Taylor) are selected by a small randomized inner search on a few validation cases to reduce manual tuning.

In general terms, the loss function condenses how far predictions deviate from data and prior knowledge into a single number; lower values indicate better agreement, and gradients guide the optimizer to reduce this number over successive iterations. [Figure 22]

```

# Build network input: [τ, normalized feed, normalized DOC, material_id]
x = torch.stack([tau, f_n.repeat(t.shape[0]), d_n.repeat(t.shape[0]), mat.repeat(t.shape[0])], 1)
vb_hat = model(x).squeeze(-1)
vb_hat = torch.clamp(vb_hat, min=0.0) # VB cannot be negative

# Data loss (only where labels are present)
mask = ~torch.isnan(vb)
L_data = F.mse_loss(vb_hat[mask], vb[mask]) if mask.any() else torch.tensor(0.0, device=DEVICE)
# Initial condition: encourage prediction at earliest time to be near zero
near0 = t == t.min()
L_ic = (vb_hat[near0]**2).mean() if near0.any() else torch.tensor(0.0, device=DEVICE)

# Physics anchor: at τ=1 the VB should be close to a reference percentile of the training data for VB at τ=1
x_tau1 = torch.stack([torch.tensor(1.0, device=DEVICE), f_n, d_n, mat]).unsqueeze(0)
vb_tau1 = model(x_tau1).squeeze()
vb_tau1 = torch.clamp(vb_tau1, min=0.0)
L_anchor = (vb_tau1 - vb_ref_t)**2

# Combine losses for this case with weights (lambdas)
losses.append(L_data + lam_ic*L_ic + lam_taylor*L_anchor)

```

Figure 22 – Loss function as sum of the three loss terms (L_{data} , L_{ic} , L_{anchor})

In the presented case there are three loss terms:

- L_{data} : This is the part of the loss that makes the model match the measurements. For every timestamp where a ground truth VB label exists, it computes the mean squared error between the predicted VB and the observed VB, then averages over those points. Minimizing this term forces the network to reproduce the actual wear curves in the dataset, while the IC and Taylor terms act as gentle physics-based regularizers to keep solutions physically plausible, especially when data are sparse or noisy.
- L_{ic} : anchors the start of each run so predicted wear is near zero at the earliest time. This prevents the network from learning curves that start with spurious nonzero or negative wear when early data are noisy or sparse.
- L_{anchor} : anchors the overall time scaling by nudging the prediction at scaled time $\tau = 1$ toward a reference wear level. This ties the learned Taylor parameters to a common, physically sensible scale across different feeds and depths of cut, reducing drift, improving identifiability, and helping the model generalize beyond the observed samples.

4.3.5 Data Loss term L_{data} and Mean Square Error on valid indices

In the code, the fit to labeled wear measurements is enforced by the mean squared error computed **only** at indices where ground truth is present. Denoting by Ω the set of valid (non-missing) labels:

$$L_{data} = \frac{1}{|\Omega|} \sum_{i \in \Omega} (\widehat{VB}_i - VB_i)^2$$

Here, **MSE** is the average of squared residuals; it penalizes larger errors more than smaller ones, which is often desirable when large deviations are especially harmful. The set Ω is the index set of available labels after masking missing values; focusing the sum on Ω prevents missing data from biasing training.

4.3.6 Initial Condition Penalty L_{ic}

The implementation encourages negligible wear at the earliest recorded time t_{min} in each case by penalizing the squared prediction at that instant:

$$L_{ic} = \mathbb{E}[\widehat{VB}(t_{min})^2]$$

In broader terms, such a **penalty** is a *soft constraint* used to encode a physically plausible boundary or initial condition without enforcing it as an exact equality. Soft constraints improve robustness to noise and outliers and stabilize training when early data are sparse, acting as a regularizer that steers solutions toward realism.

4.3.7 Anchor term L_{anchor}

In the code, a reference wear level VB_{ref} is computed from the training set as the 85th percentile of the last observed wear in each case. The 85th percentile is chosen because it is high but robust: the maximum is noisy and can be dominated by outliers or prematurely stopped runs, whereas the mean or median are too low to represent a typical near-failure level. The model is then nudged so that its prediction at the canonical scaled time $\tau = 1$ matches this reference by adding the loss.

$$L_{anchor} = (VB(\tau = 1) - VB_{ref})^2$$

This anchor reduces ambiguity between time scaling and output magnitude by tying the learned T_f to an empirically typical wear level and using $\tau = 1$ provides a common yardstick across cases and cutting conditions.

4.3.8 Training Loop

The training routine is executed end-to-end so that the neural regressor and the Taylor re-parameterized variables are updated jointly.

For each held-out case, the remaining cases constitute the training split. Feed and DOC are standardized using training means and standard deviations; these scalers are cached and reused at inference. [Figure 23]

```
# Compute normalization stats for feed/DOC
f_mean = torch.tensor(np.nanmean(train_df["feed"].values), dtype=torch.float32)
f_std = torch.tensor(np.nanstd (train_df["feed"].values)+1e-6, dtype=torch.float32)
d_mean = torch.tensor(np.nanmean(train_df["DOC"].values), dtype=torch.float32)
d_std = torch.tensor(np.nanstd (train_df["DOC"].values)+1e-6, dtype=torch.float32)
```

Figure 23 – Normalization for feed/DOC

A case wise DataLoader is constructed so that each mini-batch contains a small set of **entire cases**. For each case, time samples are sorted, and a single modal value per case is selected for feed, DOC, and material to define constant cutting settings during the forward pass. A fixed random seed is set to stabilize shuffling and weight initialization.

Training proceeds for a fixed number of epochs. In each epoch, mini-batches of cases are drawn and processed independently. Progress is monitored through periodic logging of the composite loss and the current Taylor parameters.

For each case in the mini-batch, the current Taylor parameters are first materialized through the sigmoid and softplus maps. [Figure 24]

```
n = sigmoid_range(s.a_n, *N_RANGE)
if s.use_extended:
    m = sigmoid_range(s.a_m, *M_RANGE)
    p = sigmoid_range(s.a_p, *P_RANGE)
else:
    m = torch.zeros_like(n); p = torch.zeros_like(n)
K = F.softplus(s.b_K)
return n, m, p, K
```

Figure 24 – Sigmoid and Softplus mapping for Taylor's variables

With these values and the case's constant settings (and fixed cutting speed from the dataset), the characteristic time T_f is obtained, and the dimensionless time series τ is formed by dividing the sorted physical times. The input matrix is then assembled by concatenating τ , standardized feed and DOC (broadcast along the time dimension), and the material identifier. The network evaluates this input to produce a wear trajectory $\widehat{VB}(t)$; the result is clamped from below to enforce non-negativity. Masking is applied so that only available labels contribute to the data-fit computation. [Figure 25]


```

for c in cases:
    # Convert per-case arrays and constants to tensors
    t = torch.tensor(c.t, dtype=torch.float32, device=DEVICE)
    vb = torch.tensor(c.vb, dtype=torch.float32, device=DEVICE)
    feed = torch.tensor(c.feed, dtype=torch.float32, device=DEVICE)
    doc = torch.tensor(c.doc, dtype=torch.float32, device=DEVICE)
    mat = torch.tensor(c.material, dtype=torch.float32, device=DEVICE)
    V = torch.tensor(V_CONST, dtype=torch.float32, device=DEVICE)

    # Normalize feed and DOC using training statistics
    f_n = (feed - f_mean.to(DEVICE)) / f_std.to(DEVICE)
    d_n = (doc - d_mean.to(DEVICE)) / d_std.to(DEVICE)
    # Compute Taylor time scale and scaled time  $\tau = t / T_f$ 
    Tf = taylor_Tf(V, feed, doc, n, m, p, K)
    tau = t / Tf

    # Build network input: [ $\tau$ , normalized feed, normalized DOC, material_id]
    x = torch.stack([tau, f_n.repeat(t.shape[0]), d_n.repeat(t.shape[0]), mat.repeat(t.shape[0])], 1)
    vb_hat = model(x).squeeze(-1)
    vb_hat = torch.clamp(vb_hat, min=0.0) # VB cannot be negative

    # Data loss (only where labels are present)
    mask = ~torch.isnan(vb)
    L_data = F.mse_loss(vb_hat[mask], vb[mask]) if mask.any() else torch.tensor(0.0, device=DEVICE)

```

Figure 25 – Per case forward pass

For each case, a scalar loss L is computed by combining the masked data-fit term with the two regularization terms already defined in the formulation. Per-case losses inside the mini-batch are averaged. This averaging ensures that cases contribute uniformly regardless of their number of time samples. [Figure 26]

```

# Backprop over the mean loss for the batch
batch_loss = torch.stack(losses).mean()
batch_loss.backward()
opt.step()

```

Figure 26 – PINNs backpropagation

Gradients are computed with respect to **all** learnable quantities: the network weights and the unconstrained Taylor variables. The computational graph includes the nonlinear re-parameterizations and the exponents in the characteristic time, so updates to these variables immediately modify the time scaling in subsequent iterations. A single optimizer step is then applied:

$$(\theta, a_n, a_m, a_p, b_K) \leftarrow \text{Adam} \left((\theta, a_n, a_m, a_p, b_K), \nabla L_{\text{batch}} \right)$$

Trough Adam every parameter keeps its own running stats of the gradient:

- if a parameter's gradients are large or noisy the average of recent squared gradients for parameter grows, the denominator is bigger, and its steps get **smaller** for stability
- If gradients are **small** the average of recent squared gradients for parameter stays small, giving that parameter **larger** steps so it doesn't learn too slowly

After the final epoch, the learned scalers and Taylor parameters are retained, and the model is switched to evaluation mode. Prediction on the held-out case reuses the same preprocessing and time-scaling steps, after which plotting and metric computation are carried out. This separation ensures that no training-only randomness affects the reported generalization results.

5. Results

This chapter presents the empirical evaluation of tool-wear prediction on the NASA milling dataset under a leave-one-case-out protocol. Two modelling approaches are examined:

- a purely data-driven deep neural network (DNN) baseline trained on the same inputs and splits;
- the proposed physics-informed neural network (PINN) presented in the previous chapter

For both models, preprocessing (standardization of feed and depth of cut), data partitioning, and optimization settings are kept aligned to ensure a fair comparison.

Performance is reported at case level and in aggregate using mean absolute error (MAE), error variance, and the proportions of samples with absolute error exceeding 0.05 mm and 0.10 mm; visual diagnostics (predicted trajectories versus ground truth with per-sample absolute error) are also provided. The chapter is organized as follows:

- The DNN predictions of VB are presented with quantitative metrics and illustrative plots;
- The PINN predictions are reported using the same evaluation protocol;
- A comparative analysis contrasts the two approaches across cases and overall, highlighting differences in accuracy, robustness, and error distribution, and discussing practical implications for wear monitoring.

5.1 Tool wear prediction DNN

For comparison with the PINNs model, a data-driven neural network was trained on the same dataset. The corresponding results are reported in this subsection.

Figure YY defines the multilayer perceptron used for VB regression. It builds a configurable sequence of fully connected layers of width HIDDEN and depth DEPTH, each followed by a Tanh activation and Dropout($p=0.25$) for regularization. A final linear layer maps to a single scalar output (the predicted VB). The forward method passes inputs through the

assembled `nn.Sequential`, making the network compact and easy to adjust via `in_dim`, `HIDDEN`, and `DEPTH`.

```
class MLP(nn.Module):
    """
    Small fully-connected regressor:
    - DEPTH hidden layers of width HIDDEN with Tanh activations
    - Dropout after each hidden layer
    - Final Linear layer outputs a single VB value
    """
    def __init__(self, in_dim: int, width: int = HIDDEN, depth: int = DEPTH):
        super().__init__()
        layers = []; dims = [in_dim] + [width]*depth + [1]
        for i in range(len(dims)-2):
            layers += [nn.Linear(dims[i], dims[i+1]), nn.Tanh(), nn.Dropout(p=0.25)]
        layers += [nn.Linear(dims[-2], dims[-1])]
        self.net = nn.Sequential(*layers)
    def forward(self, x): return self.net(x)
```

Figure 27 - MLP regressor architecture

The code in Figure YY trains the MLP using mini-batches from a `DataLoader` and the Adam optimizer with weight decay. During each epoch, small Gaussian noise is added to inputs and targets to improve robustness. Predictions are clamped to be non-negative to respect the physical constraint $VB \geq 0$, and the mean squared error is used as the loss. Gradients are back-propagated and parameters updated each step, with periodic logging of the training MSE; the trained model is returned at the end.

```
def train_model(X: np.ndarray, y: np.ndarray, epochs: int = EPOCHS):
    """
    Train the MLP with Adam and clamp predictions to be non-negative (VB ≥ 0)
    """
    set_seed()
    ds = RowsDataset(X, y)
    dl = torch.utils.data.DataLoader(ds, batch_size=BATCH_SIZE, shuffle=True)
    model = MLP(in_dim=X.shape[1]).to(DEVICE)
    opt = torch.optim.Adam(model.parameters(), lr=LR, weight_decay=1e-3)

    for ep in range(1, epochs+1):
        model.train(); tot=0.0
        for xb, yb in dl:
            xb, yb = xb.to(DEVICE), yb.to(DEVICE)

            af = xb + 0.01 * torch.randn_like(xb)
            zf = yb + 0.003 * torch.randn_like(yb)

            opt.zero_grad()
            pred = model(af)
            pred = torch.clamp(pred, min=0.0) # VB cannot be negative
            loss = F.mse_loss(pred, zf)
            loss.backward(); opt.step()
            tot += loss.item()*xb.size(0)

        if ep % 200 == 0 or ep == 1:
            print(f"[{ep:04d}] MSE(train)={tot/len(ds):.5f}")
    return model
```

Figure 28 - DNN Training procedure and optimization

At evaluation time, features are standardized using the training mean and standard deviation. Inside a no-grad context, the standardized tensor is forwarded through the trained model to obtain predictions. These predictions are clamped to enforce $VB \geq 0$, converted back to NumPy on the CPU, and reshaped. The results are written into a copy of the input DataFrame as a new column (VB_hat) for downstream metrics and plotting.

```
def predict_df(df: pd.DataFrame, model: nn.Module, mu: np.ndarray, sd: np.ndarray, feat_cols: List[str]) -> pd.DataFrame:
    """
    Run inference on a DataFrame:
    - z-score features with training mu/sd
    - forward through the MLP and clamp to VB ≥ 0
    - return a copy of df with an added 'VB_hat' column
    """
    X = zscore_transform(df[feat_cols], mu, sd)
    with torch.no_grad():
        vb_hat = model(torch.tensor(X, dtype=torch.float32, device=DEVICE))
        vb_hat = torch.clamp(vb_hat, min=0.0)
        vb_hat = vb_hat.cpu().numpy().reshape(-1)
    out = df.copy(); out["VB_hat"] = vb_hat
    return out
```

Figure 29 - DNN inference pipeline and post-processing

Figures 27-31 present the case-wise predictions of flank wear (VB) obtained with the data-driven DNN under the leave-one-case-out protocol. Each panel reports the ground-truth trajectory and the corresponding DNN estimate across the available sampling points; vertical bars depict the pointwise absolute error, while the horizontal axis indicates the sampling index and the vertical axis the wear in millimetres. The cases are shown individually, reflecting heterogeneous sequence lengths and wear growth rates, thereby offering a comprehensive view of the DNN's behaviour from the initial phase to the high-wear regime.

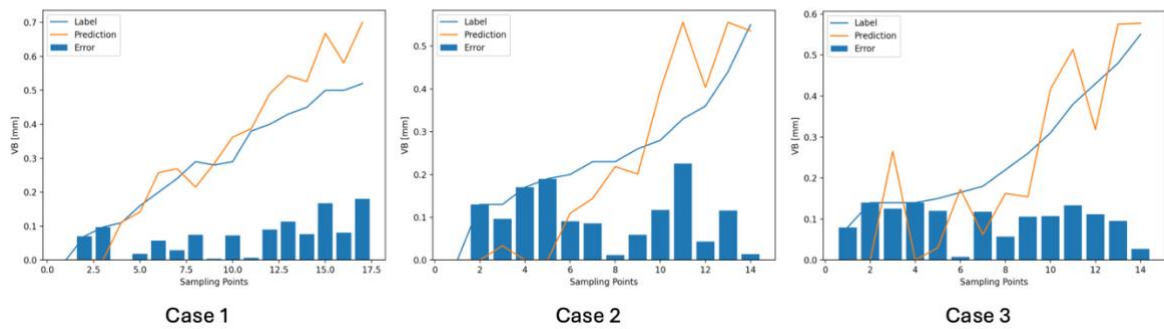


Figure 30 - Cases 1-3 DNN VB predictions

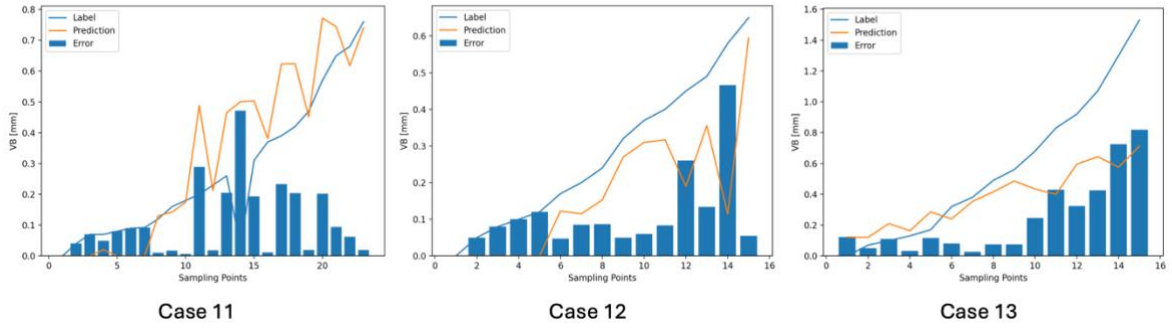


Figure 31 - Cases 4,5,7 DNN VB predictions

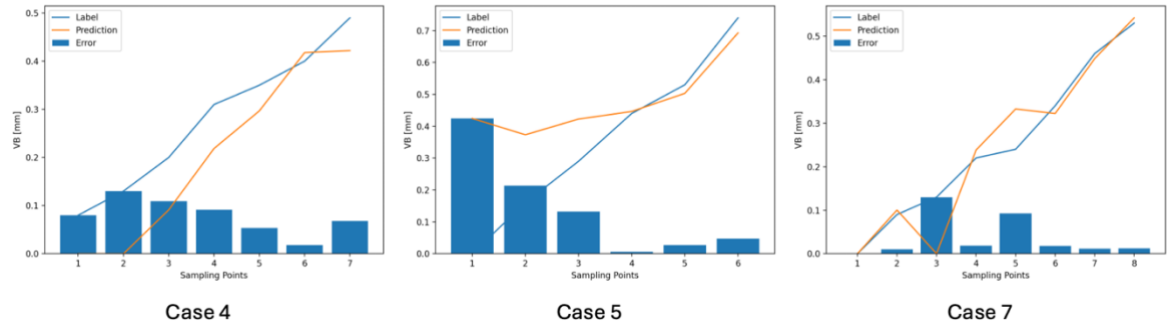


Figure 32 - Cases 8-10 DNN VB predictions

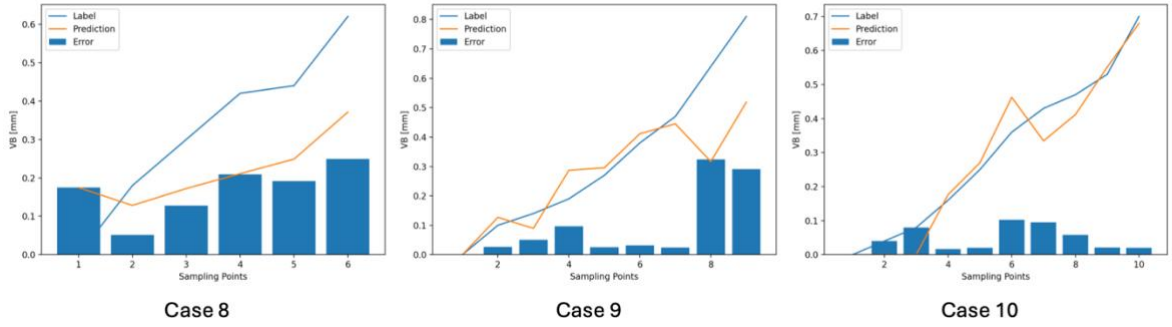


Figure 33 - Cases 11-13 DNN VB predictions

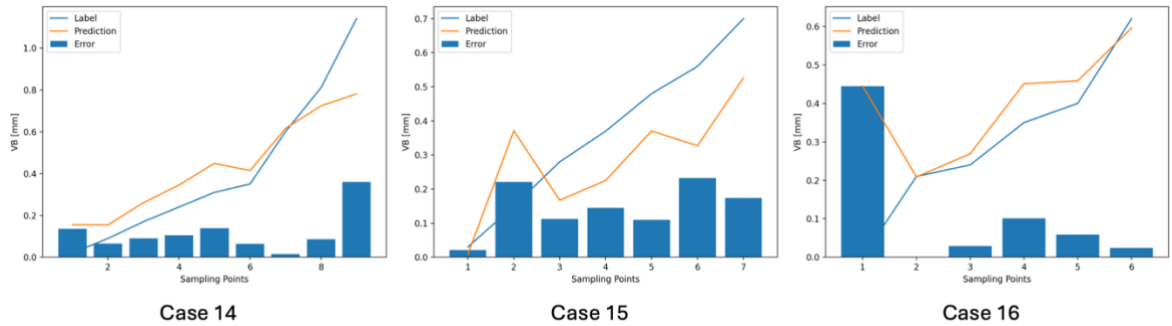


Figure 34 - Cases 14-16 DNN VB predictions

Overall, the DNN reproduces the global increasing trend of VB in most cases, with deviations that are more pronounced in rapidly rising segments and near end-of-life, where the slope steepens and absolute errors tend to concentrate. Shorter sequences appear easier

to fit but provide limited coverage of late-stage dynamics, whereas longer trajectories highlight under or over-shoot and temporal lag effects typical of purely data-driven models. These observations frame the subsequent analysis of the physics-informed approach, where time scaling and mild physical regularization are expected to stabilize predictions and reduce high-wear discrepancies.

5.2 Tool Wear prediction PINN

Figures 32-36 present the case-wise predictions of flank wear VB produced by the proposed PINN under the same leave-one-case-out protocol and preprocessing adopted for the DNN. Each panel reports, for a single case, the measured trajectory and the corresponding PINN estimate across the available sampling points; vertical bars indicate the pointwise absolute error, with the horizontal axis denoting the sampling index and the vertical axis the wear in millimetres. The layout mirrors that of the DNN figures to facilitate visual comparison, while reflecting the heterogeneity of sequence lengths and wear progressions across cases.

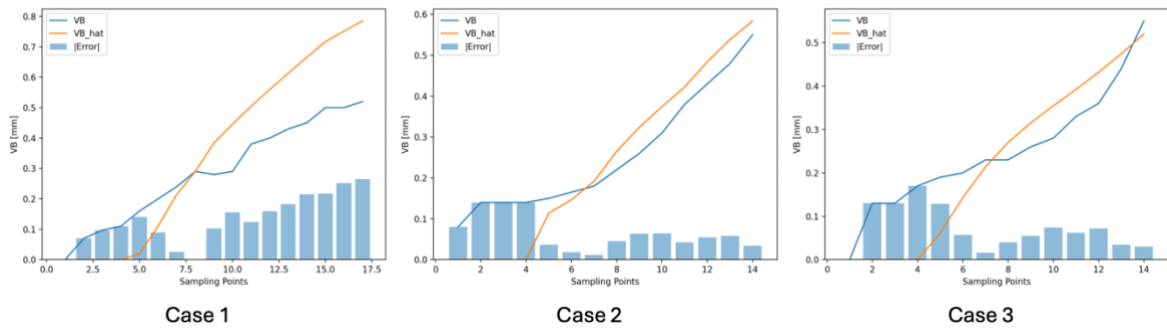


Figure 35 - Cases 1-3 PINN VB predictions

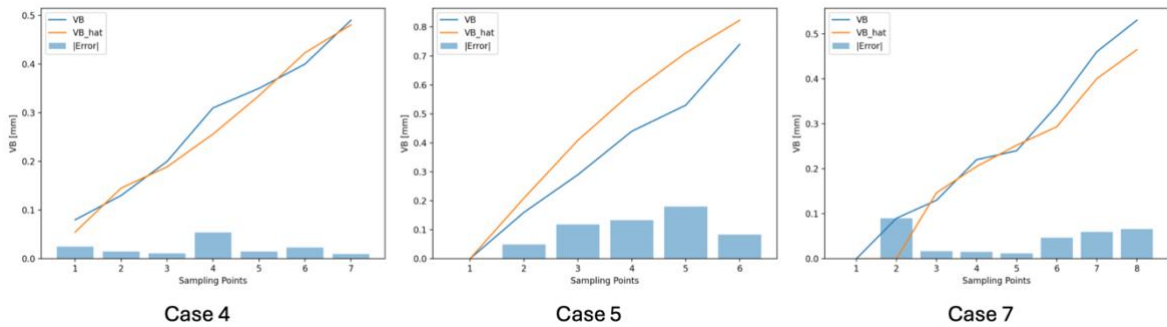


Figure 36 - Cases 4,5,7 PINN VB predictions

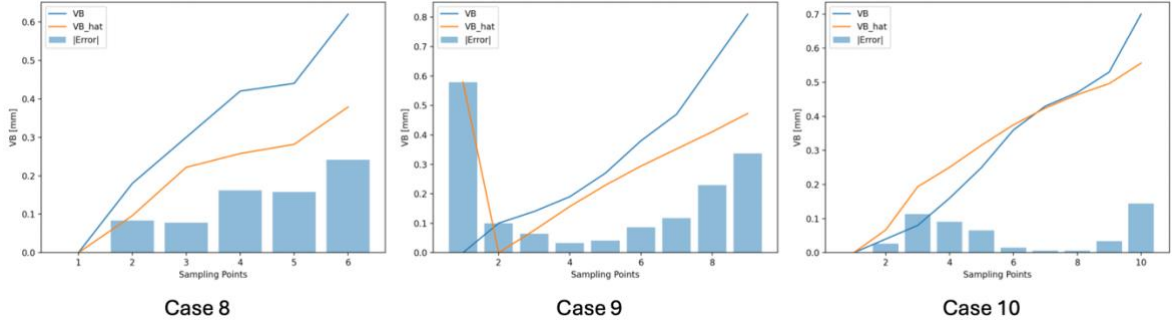


Figure 37 - Cases 8-9 PINN VB predictions

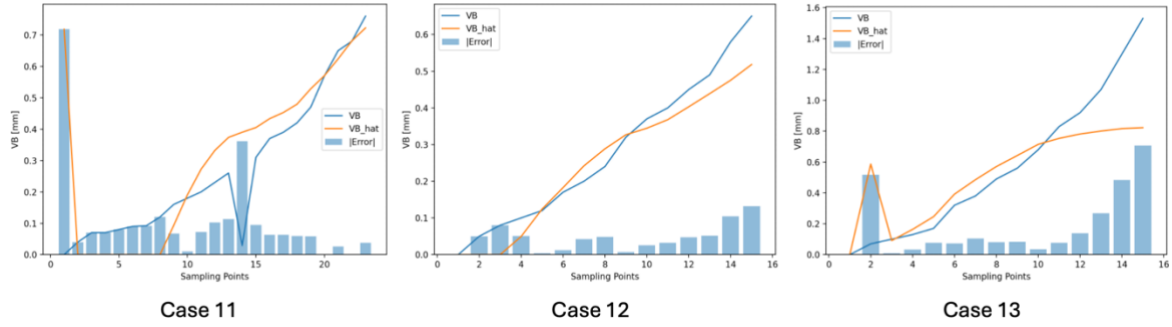


Figure 38 - Cases 11-13 PINN VB predictions

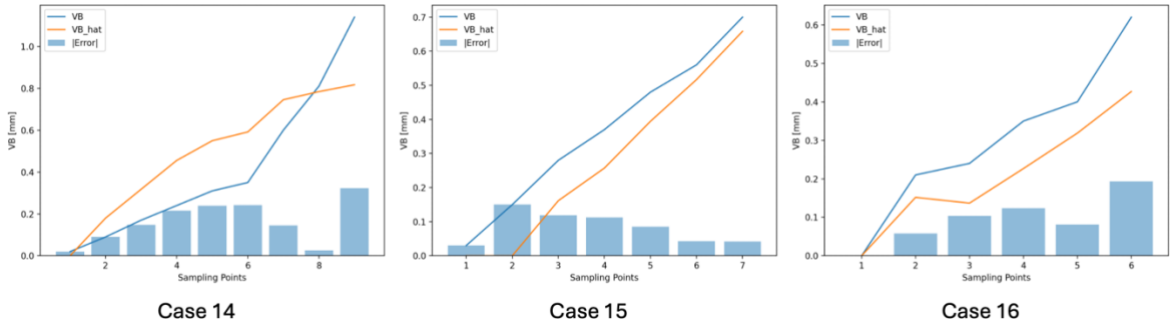


Figure 39 - Cases 14-16 PINN VB predictions

In qualitative terms, the PINN tracks the monotonic growth of VB with visibly reduced overshoot and temporal lag in the mid-to-late stages for many cases, and with improved adherence to small-wear regions at the beginning of the trajectories. Error bars tend to be more uniform along the sequence and less concentrated near steep increases, although isolated underestimation can appear where the wear accelerates abruptly. Long trajectories show smoother slope matching and fewer large deviations than typically observed for the DNN. These observations motivate the subsequent quantitative comparison, where aggregate metrics are used to assess the extent of the improvements suggested by the plots.

5.3 Results analysis

The quantitative comparison reported presents, for each of the 16 leave-one-case-out folds, four indicators computed on the DNN baseline and on the proposed PINN: **MAE**, **error variance**, and the **percentages of samples** whose absolute error exceeds **0,05 mm** and **0,10 mm**.

The accompanying bar charts present side-by-side values per case, enabling both an aggregate and a case-specific reading of performance.

Across the entire set, the PINN attains a lower **average MAE** (0.1008 vs. 0.1074), a notably lower **average variance** (0.0146 vs. 0.0202), and reduced **average threshold exceedance** rates (61.9% vs. 69.7% for 0.05 mm; 34.3% vs. 38.3% for 0.10 mm).

These figures indicate an overall improvement in accuracy and stability; nonetheless, several folds display heterogeneous behavior, which is discussed below with reference to the individual charts.

The MAE chart [Figure 40] shows that the PINN achieves a lower error than the DNN in **10 out of 16** cases, yielding a mean reduction of $\sim 6.1\%$. The most pronounced improvements appear in **Case 13** (0.2438 \rightarrow 0.1412), **Case 12** (0.1119 \rightarrow 0.0437), **Case 4** (0.0785 \rightarrow 0.0218). Deteriorations are concentrated in **Cases 1, 7, 9, 10, 11, and 14**; among these, **Case 11** (0.1078 \rightarrow 0.2713) and **Case 9** (0.0968 \rightarrow 0.1508) stand out. It is important to note that in **Cases 9 and 11** the MAE is inflated by a pronounced **spike at the first sample** visible in the PINN plots, whereas the remainder of the trajectory is closely tracked; hence MAE, being an average, over-penalizes these isolated early deviations and does not fully reflect the **overall trajectory accuracy**. Outside such edge cases, the MAE improvements align with the expectation that physics-guided time scaling mitigates late-stage overshoot observed in the DNN.

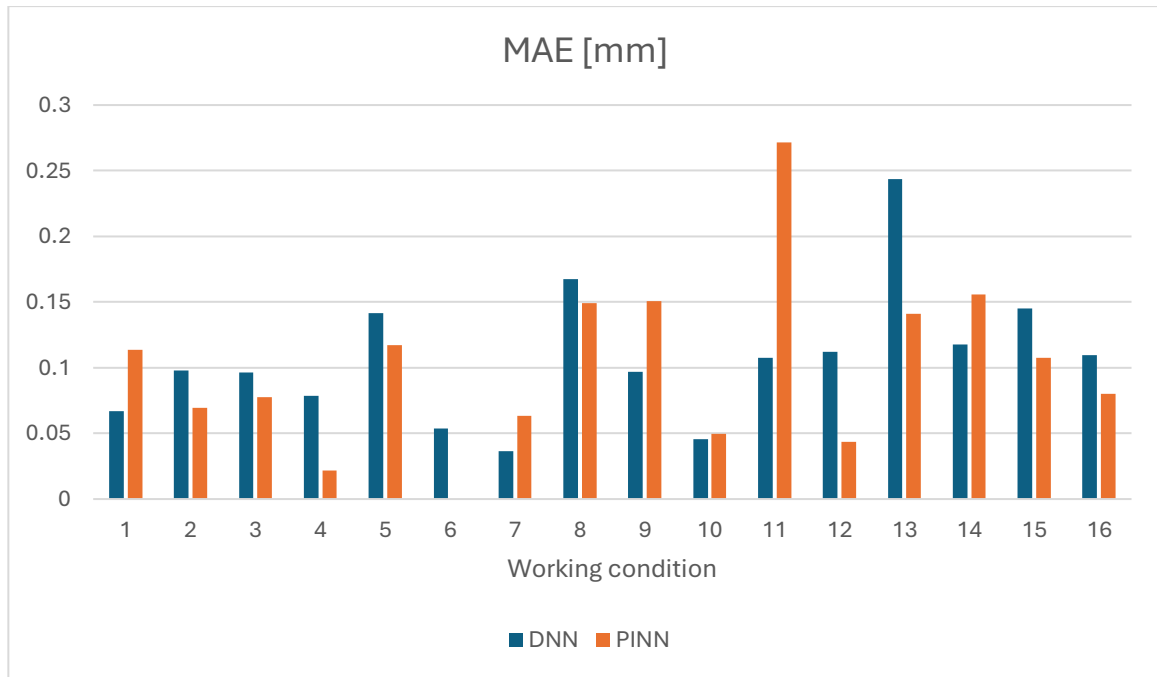


Figure 40 - Difference between DNN and PINN MAE for each working condition

The variance chart [Figure 41] highlights a stronger effect: the PINN reduces the **average residual variance by ~ 27.7%** relative to the DNN (0.0146 vs. 0.0202). Lower dispersion is observed in **10 cases**, most notably **Cases 13, 5, 16, 15, 8, and 12**, where the reduction is substantial (e.g., Case 13: 0.0907→0.0521; Case 5: 0.0328→0.0043). Variance is larger for the PINN in **Cases 9 and 11**, consistent with the single early spike that widens the residual distribution, and marginally in **Cases 1, 10, and 14**. From an inferential standpoint, the reduction in variance indicates that the PINN not only lowers average error but also **stabilizes** errors along the trajectory, diminishing the swings frequently observed in the DNN when wear accelerates.

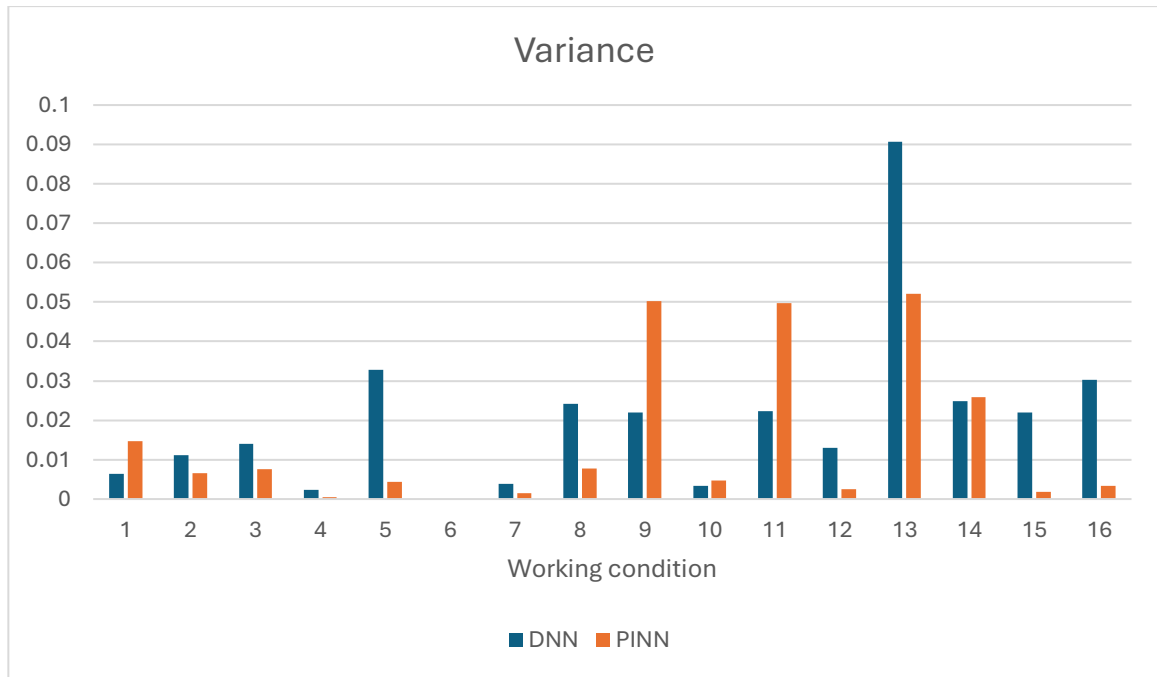


Figure 41 - Difference between DNN and PINN Variance for each working condition

For the 0.05 mm threshold, the PINN lowers the **average exceedance rate by 7.76 percentage points** (from 69.7% to 61.9%), improving **8 of 16** cases. The largest gains occur in **Case 4** (71.4%→14.3%), and **Case 12** (80%→40%); **Case 13** also shows a meaningful reduction (86.7%→60%). In contrast, higher exceedance rates are observed for **Cases 7, 5, 11, 9, 1, 16, 10, and 15**. Once again, **Cases 9 and 11** are atypical: the single large error at the first sample immediately pushes many points above the 0.05 mm threshold, thereby overstating the practical deviation over the **rest** of the trajectory. Overall, the improvement at 0.05 mm suggests that the PINN more consistently keeps **moderate absolute errors** within acceptable margins, which is operationally relevant for early warning settings. [Figure 42]

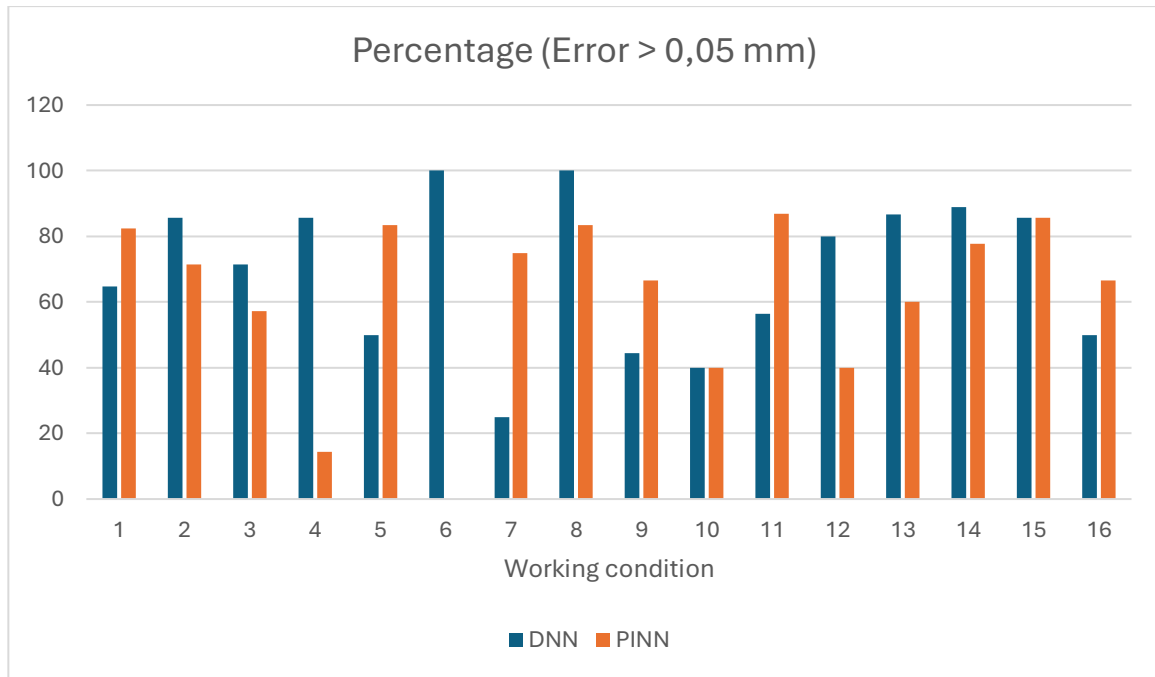


Figure 42 - Percentage of sampling points with error > 0,05 mm for DNN and PINN

At the 0.10 mm threshold, the PINN yields an **average reduction of 3.92 percentage points** (from 38.3% to 34.3%) and improves **7 of 16** cases. The most sizable gains are visible in **Case 2** (64.3%→21.4%), **Case 4** (28.6%→0%), **Case 15** (85.7%→57.1%), and **Cases 12–13** (both -20 pp). Deteriorations are concentrated in **Cases 11** (30.4%→65.2%), **1** (17.6%→41.2%), and **14** (44.4%→66.7%). As already noted, the deterioration in **Cases 9 and 11** is largely an artifact of the initial spike; threshold metrics are **sensitive to isolated outliers**, and therefore tend to understate the otherwise accurate tracking across the remaining samples. In aggregate, the shift in the 0.10 mm exceedance indicates that the PINN trims the **high-error tail**, albeit with a few outlier folds. [Figure 43]

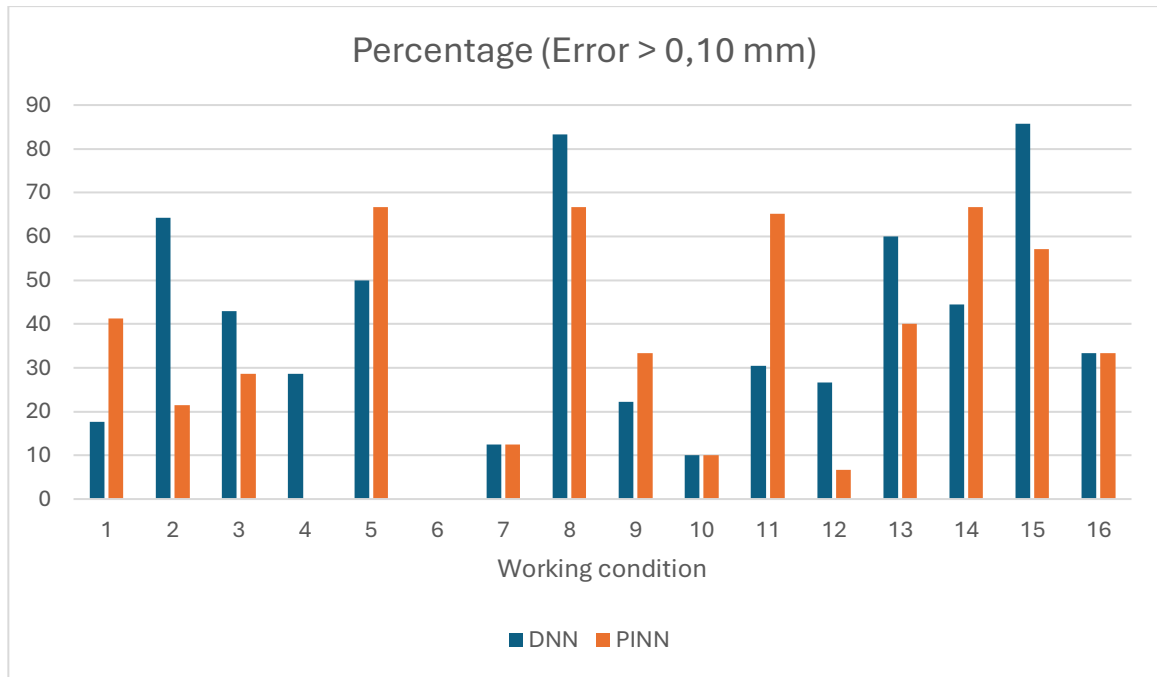


Figure 43 - Percentage of sampling points with error > 0,10 mm for DNN and PINN

Taken together, the evidence from the data indicates that the PINN delivers **more accurate and more stable** predictions than the DNN on average: lower MAE in a majority of cases, a marked reduction in error variance, and lower rates of threshold exceedance at both 0.05 mm and 0.10 mm.

The principal reservations arise in **Cases 9 and 11**, where a single **early transient** drives MAE and threshold metrics upward despite competitive trajectory tracking afterward; this behavior is visible in the corresponding PINN plots and points to sensitivity to isolated start-of-sequence anomalies rather than systematic bias.

Excluding such transients, the comparative pattern is consistent: **physics-guided time scaling dampens overshoot, narrows dispersion, and improves compliance with tolerance thresholds. These results support the adoption of the PINN as the preferred model**, while motivating future refinements that explicitly temper the influence of first-sample spikes in the evaluation.

6. Conclusions

This work set out to design, implement, and assess a physics-informed learning framework for predicting flank wear during milling, with the dual aim of improving accuracy under limited supervision and preserving physical plausibility. The proposed approach integrates an extended Taylor tool-life model into a compact fully connected regressor by learning the Taylor exponents and scale via smooth re-parameterizations, collapsing heterogeneous operating conditions through a characteristic time T_f to form a scaled input tau, and guiding training with a three-term objective (data fit, initial condition, physics anchor). The study employed the NASA milling dataset, sixteen well-controlled cases across two feeds, two depths of cut, and two materials, and adopted a leave-one-case-out protocol to ensure fair, case-level generalization tests against a like-for-like DNN baseline.

The empirical evidence indicates that the PINN is both more accurate and more stable on average. Relative to the DNN, the PINN lowers the mean MAE (0.1008 vs. 0.1074), reduces residual variance by roughly 27.7 percent (0.0146 vs. 0.0202), and decreases the proportions of samples with absolute error above 0.05 mm (61.9 percent vs. 69.7 percent) and 0.10 mm (34.3 percent vs. 38.3 percent). These gains are consistent with the qualitative plots: physics-guided time scaling mitigates mid-to-late trajectory overshoot and dampens dispersion where wear accelerates. Notable exceptions occur in Cases 9 and 11, where a pronounced first-sample spike inflates MAE and threshold exceedance despite close tracking thereafter highlighting a metric sensitivity to early outliers rather than a systematic modeling deficit.

Methodologically, three design choices proved decisive. First, scaling time by a learned T_f aligns wear trajectories across operating points, allowing the network to model shape rather than speed and thereby improving sample efficiency. Second, the initial-condition penalty regularizes early segments where labels are sparse or noisy, preventing nonphysical onset behavior. Third, the anchor at $\tau = 1$ ties magnitude to a robust, data-driven reference level and curbs ambiguity between output scale and time scaling. Together, these ingredients translate domain knowledge into soft constraints that enhance generalization without over-constraining the model.

The study has practical implications for condition monitoring and predictive maintenance. With a small number of measured VB points per case, the PINN can deliver physically credible estimates across the trajectory, potentially enabling earlier and more reliable interventions than a purely data-driven alternative. That said, two limitations warrant attention. First, the sensitivity of aggregate metrics to isolated start-of-sequence spikes

suggests adopting outlier-robust losses or temporally weighted metrics for a fairer assessment. Second, while the present formulation focuses on VB using tau, feed, depth, and material, industrial deployments would benefit from fusing richer signals and from explicit uncertainty estimates to support risk-aware decisions on tool changes.

Future work can proceed along two lines. From a modeling perspective, robust objectives and spike-aware curricula may temper the influence of early transients; hierarchical or multi-task variants could jointly learn VB and auxiliary physics to strengthen identifiability. From an architectural perspective, domain-decomposition strategies such as XPINNs and conservative PINNs could improve scalability and enforce inter-segment consistency when trajectories contain discontinuities or regime shifts. Collectively, these directions extend the demonstrated benefit of physics-guided time scaling and regularization, and position PINNs as a practical, interpretable, and data-efficient tool for next-generation tool-wear monitoring in manufacturing.

Bibliography

- [1]Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015). Deep Learning. Nature
- [2]M. Raissi, P. Perdikaris and G.E. Karniadakis (2019). Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. International Journal of Advanced Economics
- [3]George Em Karniadakis (2019). Scientific Machine Learning through Physics-Informed Neural Networks. Journal of Computational Physics
- [4]Bernard Owusu Antwi, Beatrice Adelakun, Augustine Eziefule (2024). Transforming Financial Reporting with AI: Enhancing Accuracy and Timeliness - www.researchgate.net. International Journal of Advanced Economics
- [5](2024). Six types of Neural Networks You Need to Know About - www.sabrepc.com. SabrePC Blog
- [6](2024). Deep Learning and Neural Network Trends in 2024 - www.oreilly.com. O'Reilly Media's Technology Trends 2024 report
- [7]Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi and Francesco Piccialli (2022). Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next. Journal of Scientific Computing
- [8]Christopher M. Bishop (1995). Neural Networks for Pattern Recognition. Oxford University Press
- [9]Steven L. Brunton, J. Nathan Kutz (2019). Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control. Cambridge University Press
- [10]Cortes, Vapnik (1995). Support-Vector Networks. Machine Learning (journal, Vol. 20)
- [11]George Em Karniadakis, Lu Lu, Yannis Kevrekidis, Paris Perdikaris (2021). Physics Informed machine Learning. Nature Reviews Physics
- [12]Maziar Raissi, Paris Perdikaris, Nazanin Ahmadi, and George Em Karniadakis (2024). Physics-Informed Neural Networks and Extensions. arXiv
- [13]Abhishek Jain (2024). GPUs vs CPUs (Threads, core, speed) - www.medium.com
- [14]Arthur Jacot, Franck Gabriel, Clement Hongler (2020). Neural Tangent Kernel: Convergence and Generalization in Neural Networks. NeurIPS 2018

- [15] Ameya D. Jagtap, George Em Karniadakis (2021). Extended physics-informed neural networks (XPINNs): A generalized space-timedomain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics* (Vol. 28, Issue 5, pp. 2002–2041)
- [16](2023) Different Types of Machining Process: Complete Guide - www.worthyhardware.com
- [17] Kalpakjian, S., & Schmid, S. R. (2014). *Manufacturing Engineering and Technology*. Pearson.
- [18] Groover, M. P. (2010). *Fundamentals of Modern Manufacturing: Materials, Processes, and Systems*. Wiley.
- [19] Ulutan, D. and Özel, T. (2011). Machining-induced surface integrity in titanium and nickel alloys: A review.
- [20] Teti, R., Jemielniak, K., O'Donnell, G. and Dornfeld, D. (2010). Advanced monitoring of machining operations.
- [21] Ezugwu, E. O. (2005). Key improvements in the machining of difficult-to-cut aerospace superalloys.
- [22] Denkena, B. and Biermann, D. (2014). Cutting edge geometries.
- [23] Altintas, Y. (2012). *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design*.
- [24] Taylor F. (1907). *On the Art of Cutting Metals*.
- [25] Shokrani, A., Dhokia, V. and Newman, S. T. (2012). Environmentally conscious machining of difficult-to-machine materials with regard to cutting fluids.
- [26] Niță, B., et al. (2024). Review regarding the influence of cryogenic milling on machinability and wear.
- [27] Z. Palmai (2013). Proposal for a new theoretical model of the cutting tool's flank wear.
- [28] K. Goebel, A. Agogino (2007). *NASA Milling Data Set*
- [29] Jiaqui Hua, Yingguang Li, Changqing Liu, Peng Wan, Xu Liu (2024). Physics-Informed Neural Networks With Weighted Losses by Uncertainty Evaluation for Accurate and Stable Prediction of Manufacturing Systems.
- [30] www.revelationmachinery.com
- [31] Yilin Li, Jinjiang Wang, Zuguang Huang, Robert X. Gao (2022). Physics-Informed Meta Learning for Machining Tool Wear Prediction.

- [32] Qian Yang, Krishna R. Pattipati, Utsav Awasthi, George M. Bolas (2022). Hybrid Data-Driven and Model-Informed Online Tool Wear Detection in Milling Machines.
- [33] Pengcheng Wu, Haicong Dai, Yufeng Li, Yan He, Rui Zhong, Jinsen He (2022). A Physics-Informed Machine Learning Model for Surface Roughness Prediction in Milling Operations.
- [34] Shi Zeng, Dechang Pi (2023). Milling Surface Roughness Prediction Based on Physics-Informed Machine Learning.
- [35] M. Hossein Rahimi, Hoai Nam Huynh, Yusuf Altintas (2021). On-Line Chatter Detection in Milling with Hybrid Machine Learning and Physics-Based Model.
- [36] Dazhong Wu, Connor Jennings, Janis Terpenney, Robert X. Gao, Soundar Kumara (2017). A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests.
- [37] Dazhong Wu, Connor Jennings, Janis Terpenney, Robert X. Gao, Soundar Kumara (2017). Data-Driven Prognostics Using Random Forests: Prediction of Tool Wear.
- [38] Mengdan Qiao; Yang Yang; Ningbin Zhu; Yi Wang; Sikang Li; Qingfeng Cao; Yanghui Hou; Bo Qian (2025). A physics-informed neural network integrating physical mechanisms and experimental data for predicting VOCs emission rates in machining workshops
- [39] Xiaohui Fang; Qinghua Song; Xiaojuan Wang; Zhenyang Li; Haifeng Ma; Zhanqiang Liu (2025). An intelligent tool wear monitoring model based on knowledge-data-driven physical-informed neural network for digital twin milling
- [40] K. Goebel (1996). Management of Uncertainty in Sensor Validation, Sensor Fusion, and Diagnosis of Mechanical Systems Using Soft Computing Techniques
- [41] Karel Šramhauser, Nataša Náprstková, Jan Svianteck, Dana Stančková, Nguyen Van Tuong, Jan Novotný (2022). Analyses of Tool Wear and Chip Type for Different Coated Carbide Inserts in Turning Hardened 1.6582 Steel.
- [42] Muhammad Younas, Syed Husain Imran Jaffery, Mushtaq Khan, Aftab Khan (2019). Tool Wear Progression and Its Effect on Energy Consumption in Turning of Titanium Alloy (Ti-6Al-4V).
- [43] Denkena, B. and Biermann, D. (2014). Cutting edge geometries. CIRP Annals - Manufacturing Technology.
- [44] Kitagawa, T., Kubo, A. and Maekawa, K. (1997). Temperature and wear of cutting tools in high-speed machining. Wear.
- [45] Shaw, M.C. (2005). Metal Cutting Principles. Oxford University Press.

- [46] Johansson, D. and Pejryd, L. (2017). Assessment of commonly used tool life models in metal cutting. *Procedia Manufacturing*.
- [47] Jinjiang Wang, Yilin Li, Rui Zhao, Robert X. Gao (2020). Physics guided neural network for machining tool wear prediction.
- [48] Fang, X., Song, Q. and Wang, X. (2025). An intelligent tool wear monitoring model based on knowledge-data-driven physics-informed neural network for digital twin milling.
- [49] Zhu, M., Ren, L., Li, J. and Zhou, Q. (2023). Intelligent milling tool wear prediction model based on transformer informed by physics.
- [50] Wang, Y., Chen, X., Liu, Z. and Zhao, Y. (2023). Physics-informed deep learning for tool wear monitoring.

Acknowledgments

Desidero esprimere la mia sincera gratitudine alla Prof.ssa e Relatrice Giulia Bruno per la guida attenta, la disponibilità costante e i preziosi suggerimenti che hanno reso possibile questo lavoro. Un sentito grazie anche ai Proff. Gabriele Antal e Valentino Razza per il supporto, gli spunti critici e l'incoraggiamento lungo tutto il percorso di ricerca.