

Predicting Surface Properties from Atomic Force Microscopy Nanoindentations using Artificial Neural Networks

Luís Rodrigues Lopes Barros Pacheco



Master in Mechanical Engineering

Supervisor: Dr. João Pedro Sousa Ferreira

Cosupervisor: Prof. Dr. Marco Paulo Lages Parente

July 7, 2023

Predicting Surface Properties from Atomic Force Microscopy Nanoindentations using Artificial Neural Networks

Luís Rodrigues Lopes Barros Pacheco

Master in Mechanical Engineering

July 7, 2023

Resumo

Sendo as propriedades dos materiais fundamentalmente definidas pela sua estrutura atómica, o estudo das propriedades mecânicas tem vindo a evoluir para a nanoescala. A Microscopia de Força Atómica (AFM) é um dos principais métodos quando se trata de obter imagens de uma amostra, mas é também uma técnica poderosa para estudar as propriedades nanomecânicas de uma vasta gama de materiais, bem como para caracterizar interacções ao nível do piconewton. Com as nanoindentações por AFM, as propriedades superficiais em estudo podem ser obtidas através da aplicação de modelos de contacto adequados, às curvas de força-indentação. Dado o elevado número de pontos em cada curva e a necessidade típica de analisar milhares de curvas, o processo de selecção de um modelo de contacto e a sua aplicação para inferir propriedades mecânicas através dos dados das mesmas pode ser uma tarefa difícil e morosa. Consequentemente, estão a ser exploradas alternativas a este procedimento e tem sido demonstrado que uma abordagem baseada em modelos de *Machine Learning* (ML) pode ser de grande utilidade para prever propriedades materiais relevantes, com dados AFM.

Neste trabalho, foram criados dois modelos de regressão utilizando Redes Neurais Artificiais (ANN), para prever o módulo de Young e a energia de adesão a partir de nanoindentações AFM, que foram divididas em curvas de aproximação e de afastamento. Para fins de aprendizagem, foram gerados dados sintéticos, utilizando dois modelos de contacto. As curvas de aproximação foram criadas com o modelo de contacto Hertz e utilizadas para treinar um *Multilayer Perceptron* (MLP), prevendo o módulo de Young, enquanto a teoria de contacto de Johnson-Kendall-Roberts (JKR) foi o suporte para produzir curvas de afastamento, que treinaram um segundo modelo MLP, que não só previu o módulo de elasticidade, mas também a energia de adesão.

Os dados sintéticos foram divididos em conjuntos de treino, validação e teste, tendo sempre em conta a sua estratificação com base nas variáveis a serem previstas. Utilizando a biblioteca PyTorch para construir e treinar o modelo, foram definidos e aperfeiçoados os principais hiperparâmetros, recorrendo ao *framework* de optimização Optuna. O primeiro modelo foi testado com sucesso em curvas experimentais de nanoindentações AFM e o segundo apresentou resultados promissores nos dados sintéticos. Por fim, as previsões de curvas reais foram mapeadas em superfícies tridimensionais, para ilustrar uma expansão adicional do trabalho apresentado, que permite não só inferir as propriedades superficiais a partir de curvas individuais, mas também compreender a sua distribuição na superfície da amostra.

Ao longo deste projeto foram apresentadas previsões com elevado grau de precisão e de uma forma computacionalmente eficiente, validando o potencial de uma abordagem de *Deep Learning* para explorar nanoindentações AFM e motivando assim o desenvolvimento futuro do trabalho apresentado.

Abstract

The study of mechanical properties has been trending towards the nanoscale, as material properties are fundamentally defined by their atomic structure. Atomic Force Microscopy (AFM) is one of the main methods when it comes to imaging a sample, but it is also a powerful technique to study the nanomechanical properties of a wide range of materials, as well as to characterize interactions at the piconewton level. With AFM nanoindentations, fitting the force-indentation curves obtained for each sample with a suitable contact model, allows to acquire the properties of interest. Given the high number of points in each curve and the typical need to analyse thousands of curves, the process of selecting a contact model and applying it to fit the data, can be a challenging and time-consuming task. As a result, alternatives to this procedure are being explored and it has been shown that an approach based on Machine Learning (ML) models can be of great use to predict relevant material properties, from AFM analysis.

In this work, two regression models using Artificial Neural Networks (ANN) were created, to predict Young's modulus and surface energy from AFM nanoindentations, which were divided into approach and withdraw curves. For learning purposes, synthetic data was generated, using two contact models. Approach curves were created with Hertz contact and used to train a Multi-layer Perceptron (MLP), forecasting the Young's modulus, while JKR contact was the support for producing withdraw curves, that trained a second MLP model, that not only predicted the elastic modulus, but also the surface energy.

Synthetic data was split into training, validation and test sets, always accounting for target stratification. Employing the PyTorch framework to build and train the model, we set and refined key hyperparameters, based on implementing the optimization framework Optuna. The first model was successfully tested with experimental curves from AFM nanoindentations and the second presented promising results on the synthetic data. At last, the predictions from real curves were mapped into three-dimensional surfaces, to illustrate a further expansion of the current framework, where it allows not only to infer the surface properties from individual curves, but also understand its distribution over the sample's surface.

Our framework provided accurate predictions, in a computationally efficient way, thus validating the potential of a Deep Learning approach to explore AFM nanoindentations and motivating the further development of the presented work.

Agradecimentos

Ao meu orientador, Doutor João Pedro Sousa Ferreira, por toda a motivação dada ao longo deste trabalho, pelo brio colocado em todas as sugestões e por toda a disponibilidade demonstrada. Agradeço ainda pelas oportunidades proporcionadas e pelo incentivo constante a querer fazer mais e melhor. Ao meu coorientador, Professor Marco Paulo Lages Parente, por toda a confiança depositada.

Aos meus amigos, Daniel, Diogo, Eduardo, Francisco, Jorge, José, Rodrigo e Tiago, pelas memórias e experiências que levarei sempre comigo, e a todos aqueles com quem tive o privilégio de partilhar os últimos cinco anos.

À minha família, em especial aos meus avós, pelo apoio incondicional e por serem um exemplo de trabalho e dedicação.

À minha mãe, a quem devo tudo o que consegui até hoje, o meu maior obrigado. À minha irmã, pela boa disposição constante e pelo vasto leque musical que me introduziste nas centenas de viagens em hora de ponta. À Beatriz, por todo o apoio nos momentos mais difíceis, por acreditares sempre em mim e por seres a minha grande motivação.

Luís Pacheco

“Voando a máquina,
todo o céu será música.”

José Saramago, in *Memorial do Convento*

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation and Objectives	3
1.3	Thesis outline	3
2	Atomic Force Microscopy	5
2.1	Introduction	5
2.2	Working Principle and Instrumentation	7
2.3	Operating Modes	10
2.3.1	Topographic Modes	11
2.3.2	Non-Topographic Modes	12
2.4	Mapping of Mechanical Properties	15
2.5	Applications	18
3	Contact Mechanics	21
3.1	Introduction	21
3.2	Hertz Theory	22
3.2.1	Basic Principles	22
3.2.2	Overcoming Hertzian Theory Limitations in AFM Analysis	24
3.3	JKR theory	26
4	Machine Learning	31
4.1	Introduction	31
4.2	Train-Validation-Test Framework	33
4.3	Artificial Neural Networks	35
4.3.1	Perceptron	35
4.3.2	Artificial Neuron	37
4.3.3	Loss Functions	40
4.3.4	Backpropagation	40
4.3.5	Optimization Algorithms	43
4.4	Implementation in PyTorch	45
5	Data Generation and Model Development	51
5.1	Synthetic Data Generation	51
5.2	Development of the Hertz MLP	58
5.2.1	Stratification and Split Ratio	59
5.2.2	Activation and Loss Functions	61
5.2.3	Optuna Implementation	63

5.2.4	Evaluating the Models on Synthetic Data	71
5.2.5	Hertz MLP Concluding Remarks	73
5.3	Development of the JKR MLP	74
5.3.1	Stratification and Split Ratio	75
5.3.2	Activation and Loss Functions	78
5.3.3	Optuna Implementation	79
5.3.4	Evaluating the Final Models	84
5.3.5	JKR MLP Concluding Remarks	86
6	Testing on Experimental Data and Surface Mapping	89
6.1	Introducing the Dataset	89
6.2	Preprocessing Experimental Data	91
6.3	Hertz MLPs Performance on Experimental Data	92
6.4	Generating Stiffness Maps	96
6.4.1	Creating 3D Surfaces	96
6.4.2	From Height Distribution to Stiffness Maps	97
6.5	Predicting Stiffness Maps with Hertz MLP	97
7	Conclusions and Future Work	103
7.1	Conclusions	103
7.2	Future Work	104
	References	105
A	Implementing Optuna in Pytorch	111
A.1	Defining the Model	111
A.2	Defining the Objective Function	112

List of Figures

2.1	Comparison of imaging ranges for different microscopy techniques.	6
2.2	AFM Force-Indentation example curve, obtained for a smooth muscle cell.	7
2.3	Schematic view of the AFM method (14).	8
2.4	Views of a typical AFM probe (47).	9
2.5	Photodetector identifying cantilever deflection and torsion. Adapted from (12). .	12
2.6	A model of an ideal AFM Force-Distance curve. A - Probe's initial position; B - Maximum indentation point; C - Point of maximum adhesion force. Adapted from (12).	13
2.7	Illustration of the cantilever deflection in different AFM stages.	14
2.8	Experimental setup to obtain F-D or F-I AFM curves, where the dashed cantilever represents its original/undeformed shape. z_c represents the cantilever deflection and δ the indentation depth.	15
2.9	Example of the raw output data obtained from nanoindentation in an undeformable surface.	16
2.10	Young's modulus map from a cortical bone sample, obtained with AFM nanoindentations (5).	19
3.1	Scheme of two spheres in contact.	22
3.2	Relation between force and indentation depth, computed through Hertz theory-based approximation in Equation 3.15.	26
3.3	Scheme of two spheres in contact, with (R_{c1}) and without (R_{c0}) surface forces. . .	27
3.4	Relation between force and indentation depth, computed through JKR theory-based approximation in Equation 3.28, setting the Young's modulus as an unitary constant.	29
4.1	Illustration of the most important AI subsets for this framework, where ANNs stands for Artificial Neural Networks.	32
4.2	Example of the evolution of training and validation loss throughout the epochs. .	34
4.3	Example of an ANN composed only of dense layers, with two hidden layers. Adapted from (29).	36
4.4	Scheme of a perceptron that takes 3 inputs.	37
4.5	Activation of an artificial neuron. Adapted from (29).	38
4.6	Activation function plots (blue) and their gradients (dashed orange). For illustration purposes, the Leaky ReLU constant is set to 0.1, instead of the usual 0.01. . .	39
4.7	Comparison between loss functions, considering $t = 2$ for the Huber Loss.	41
4.8	Simple ANN with only one node in each layer. Only the nodes in layers $l - 1$, l and in the output layer L are represented.	41
4.9	Flattening the inputs.	47

5.1	Unrealistic representation of AFM nanoindentation curves ($E \in [0.1-10]$ kPa and $\gamma \in [10-100]$ $\mu\text{J}/\text{m}^2$). The lines represent the mean and the bands the standard deviation.	54
5.2	Distribution of the synthetic approach and withdraw curves for the initial dataset.	54
5.3	Initial distribution of the material parameters E and γ	56
5.4	Synthetic nanoindentation curves in the initial dataset, for distinct E and γ configurations.	57
5.5	Distribution of the target variable over the different sets, with and without stratification.	59
5.6	Distribution of the relative error for the testing samples.	60
5.7	Test Loss (MSE) obtained for 3 different split ratios (Train/Test/Validation) and 20, 40 and 60 epochs. For each split proportion-epoch pair, 3 random splits were tested.	61
5.8	Error distribution when using the Huber Loss function and the Leaky ReLU activation, alongside the base model hyperparameters.	62
5.9	Optuna framework scheme with pruning, where N is the number of epochs.	63
5.10	Relative error distribution for the 2 models analysed.	66
5.11	MSE loss obtained for a batch size of 16 and 32, over several trials of this analysis.	68
5.12	MSE loss obtained from using 16 or 32 nodes in the last hidden layer, across the trials for all studies of analysis 3.	70
5.13	Relation between Loss and Learning Rate for different trials over the three studies.	71
5.14	Relative error distribution for the models that behaved worst and best on the synthetic testing set, respectively.	73
5.15	Actual and predicted synthetic curves for different Young's moduli with relative errors near 2%, 10% and greater than 20%.	74
5.16	Distribution of the variable γ over the different sets, with and without stratification.	76
5.17	Relative error distribution for the material parameters in the testing set, with and without data stratification.	77
5.18	MSE loss obtained in the testing set with different split ratios and number of epochs, for both material parameters (E at the left and γ at the right).	77
5.19	Average relative error obtained for the 5 test folds (initial test set and 4 additional folds), using MAPE and MSE as loss functions.	79
5.20	Loss distribution for trials with a batch size of 16 and 32. Trials with losses higher than 0.2 were eliminated for likely being associated with early pruning.	81
5.21	Loss as a function of the learning rate in Analysis 4.	83
5.22	Relative error obtained for the 5 test folds, with the three best performing models.	85
5.23	Predicted and actual retraction curves for different relative errors of E	87
5.24	Predicted and actual retraction curves for different relative errors of γ	87
6.1	Examples of experimental approach curves in the available AFM nanoindentations data.	90
6.2	Young's modulus distribution over the experimental curves.	90
6.3	Examples of experimental approach curves that were fitted with an r^2 below 0.9.	91
6.4	Mean synthetic and experimental curves, with errorbar representing the standard deviation.	92
6.5	Comparison between the relative error distribution, when selecting experimental curves with different minimum r^2 fitting values.	94
6.6	Experimental curves with $r^2 > 0.99$ with a high relative error.	95
6.7	Comparison between predicted and experimental curves.	95

6.8	Initial synthetic surface maps.	98
6.9	Real and predicted stiffness maps for the smooth surface.	98
6.10	Real and predicted stiffness maps for the rough surface.	99
6.11	Relative error maps for the smooth and rough surfaces, respectively.	99
6.12	Relative error distribution for the predictions in the smooth (left) and rough (right) surfaces.	100
6.13	Actual and predicted curves for 3 points identified in the smooth surface.	100

List of Tables

5.1	Initial variables definition for synthetic data generation.	53
5.2	Ranges of values for material properties.	55
5.3	Hyperparameters of the baseline Hertz MLP model.	58
5.4	Initial activation and loss functions, optimization algorithm and split ratios for the Hertz MLP.	58
5.5	Average test loss and error values for the model with and without stratification. . .	60
5.6	Error values for different loss and activation functions.	62
5.7	Hyperparameters for the Hertz MLP first analysis.	65
5.8	Best trials for the study with Leaky ReLU as the activation function.	66
5.9	Best trials for the study with ReLU as the activation function.	66
5.10	Hyperparameters being optimized in the second analysis of the Hertz MLP. . . .	67
5.11	Best trials in each of the 5 studies performed in Analysis 2.	67
5.12	Hyperparameters being optimized in the Analysis 3.	69
5.13	Best trials for each study in Analysis 3.	69
5.14	Allowed range for the Learning Rate in Analysis 4.	70
5.15	Best trials for the 3 studies in Analysis 4.	71
5.16	Hyperparameters of the final Hertz MLPs. All used Adam as the optimization algorithm, while Leaky ReLU and Huber Loss were applied in all, with the exception of the base model (ReLU and MSE).	72
5.17	Performance of the models on the synthetic test set.	72
5.18	Hyperparameters of the baseline JKR MLP model.	75
5.19	Initial activation and loss functions, optimization algorithm and split ratios for the JKR MLP.	75
5.20	Average test error values for both target variables, with and without stratification. .	76
5.21	Error values for different loss and activation functions.	78
5.22	Hyperparameters for the JKR MLP first analysis.	79
5.23	Best trials for the studies in the first analysis of the JKR MLP.	80
5.24	Hyperparameters being optimized in the second analysis of the JKR MLP.	80
5.25	Best trials in each of the 5 studies performed in Analysis 2.	81
5.26	Hyperparameters being optimized in the third analysis of the JKR MLP.	82
5.27	Best trials in each of the 5 studies performed in Analysis 3.	82
5.28	Best trials for the 3 studies regarding Model 4a).	83
5.29	Best trials for the 3 studies regarding Model 4b).	83
5.30	Hyperparameters of the final JKR MLPs, highlighting the model with the best performance - Model 4a). The optimizer (Adam), loss (MSE) and activation (ReLU) were common to all the models.	84
5.31	Average performance of the JKR MLP models on the 5 test folds, regarding material parameter E	84

5.32	Average performance of the JKR MLP models on the 5 test folds, regarding material parameter γ	85
6.1	Number of curves and Young's modulus' statistics in the final experimental dataset.	92
6.2	Performance of the models on the experimental test set. Note that the loss value in the Base Model was computed with the MSE, while Huber Loss was applied for the remaining. The best performing model is highlighted.	93

List of Symbols

Abbreviations

AFM	Atomic Force Microscopy
AI	Artificial Intelligence
ANN	Artificial Neural Network
BGD	Batch Gradient Descent
DL	Deep Learning
DMT	Derjaguin-Muller-Toporov
F-D	Force-Distance
FEM	Finite Element Method
F-I	Force-Indentation
HL	Hidden Layers
HPD	Height Probability Distribution
JKR	Johnson-Kendall-Roberts
LR	Learning Rate
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NN	Neural Network
PS	Power Spectrum
ReLU	Rectified Linear Unit
SFM	Scanning Force Microscopy
SGD	Stochastic Gradient Descent
SPM	Scanning Probe Microscopy
STM	Scanning Tunneling Microscopy

Variables

α	Cone half angle
γ	Adhesion energy
δ	Indentation depth
ε	Percentage relative error of an instance
$\bar{\varepsilon}$	Mean percentage relative error
η	Learning rate

θ	Model's parameters
μ	Tabor parameter
ν	Poisson ratio
Φ	Error at a given neuron
A_{max}	Contact area at maximum indentation
a	Node output
B	Fitting constant
b	Bias terms of an ANN
d	Constant that reflects material properties
E	Young's (or elastic) modulus
E^*	Effective Young's modulus
F_{ad}	Adhesion force
F_{max}	Maximum force
F_{spring}	Cantilever spring force
f	Activation function
H	Hardness
K	Variation of the effective Young's modulus
k	Spring constant
\mathcal{L}	Loss function
p	Pressure
p_0	Maximum contact pressure
R	Radius
R_c	Contact radius
R_t	Transition radius for a sphero-conical tip indenter
r	Radial distance from the center of contact
S	AFM system sensitivity
s	Equilibrium separation between two surfaces
T	Number of epochs
ΔV	Photodetector voltage output
W_a	Work of adhesion
w	Weight parameters of an ANN
y	Actual output
\hat{y}	Predicted output
\bar{y}	Mean of the actual outputs
Z	Correction factor
z	Weighted inputs of a neuron (pre-activation)
z_0	Zero indentation point
z_c	Cantilever deflection
z_p	Cantilever vertical displacement (z-piezo displacement)

Chapter 1

Introduction

1.1 Context

The last decades have brought significant advancements in the field of nanotechnology at various levels. Several areas of research have greatly benefited from these improvements, and mechanical engineering is no exception, with the study of materials at their nanoscale being substantially leveraged. Atomic Force Microscopy (AFM) is one of the main techniques responsible for these studies. Urging from the need of visualizing and manipulating materials at their lower scale, this method was established in the late 20th century, following the concept of Scanning Probe Microscopy. It is based on a small probe, usually consisting of a cantilever with a tip that can have several geometries attached to its end. From the interaction of the probe with the sampled surface, whether involving contact or not, the target properties can be inferred.

High-resolution and three-dimensional images of the surface topography at the nanoscale can be generated by this probe-sample interaction. Furthermore, due to its wide range of operating modes, not only the surface height can be studied, but also key surface properties that contribute to understanding the behaviour of materials at increasingly smaller scales. It is the case of AFM nanoindentations, that consist in approaching the probe towards the sample and indenting it for a defined threshold, followed by withdrawing the probe until it detaches from the sample or reaches its original position.

Along this ramp cycle, the deflection of the probe cantilever is recorded as a function of the probe's displacement, which can later be translated into Force-Indentation (F-I) curves. By fitting these curves with appropriate contact models, as Hertzian or Sneddon theories for the approach stage and JKR or DMT for the retraction cycle, the surface properties are finally obtained. Furthermore, if there is knowledge about the location in the sample where each F-I curve was obtained, it is possible to create surface maps for each measured property. However, the curve-fitting process can be quite time-consuming.

Regarding approach curves, they play a crucial role in inferring the elastic modulus of the sample. On the other hand, retraction data offers essential insights on its adhesion properties, which are particularly relevant in low stiffness materials, as biological cells, where they play a

major role in processes such as cell proliferation and migration, and sudden changes in these properties can lead to diseases as cancer or osteoporosis (9).

Another field that has witnessed remarkable advancements in recent times is Artificial Intelligence (AI) and its subset of Machine Learning (ML), taking advantage of the exponential increase in computational power. ML algorithms can make data-based decisions autonomously, without relying on predefined programming instructions. Hence, the availability of expanding datasets in various domains has led to the development of diverse and adaptable models. These models are nothing but mathematical representations that aim to capture the input features from the data provided and associate them with the corresponding outputs. While the process of designing and optimizing these models can be complex, when concluded they demonstrate fast and consistent performance.

Being able to adapt to different sorts of data, allows ML to be applied across various fields, from medicine and engineering to finance or marketing. So, it only makes sense that it can also be used to boost techniques as AFM, mainly in the postprocessing stage. For instance, as many contact models only assume linear elastic behaviour, the fact that ML models can easily capture nonlinearity is a great opportunity to overcome such limitations.

In recent years, several publications have introduced various approaches that utilize ML methods to assist in diverse types of AFM analyses. A ML approach based on Artificial Neural Networks (ANNs) is used in (57) to classify bladder cancer cells into different grades, based on cellular mechanical properties obtained with AFM. For an application more related to AFM intrinsic properties, Convolutional Neural Networks (CNNs) models have been developed to determine the tip sharpness directly from indentation images (55). A different study (2) has presented a Quasi-Recurrent Neural Network to identify the coupling of vibrating modes in dynamic AFM (intermittent contact between probe and sample). At last, ML regression models were used to predict the elastic modulus based on nanoindentation curves, without having to fit them with a contact model, in the work presented in (30), where the ML models were trained with experimental F-I curves from AFM analyses. This last work shares some similarities with the current project, in the sense that they both focus on determining the sample's stiffness without contact model fitting. Nevertheless, the work in (30) doesn't cover the development of Deep Learning strategies and it requires experimental data to train the models, unlike the current framework.

Despite the several ML applications on the AFM field, there is not yet a Deep Learning-based approach to determine surface properties, related to the elastic modulus and to the sample's adhesion, from both approach and retraction curves, without requiring the model to be trained on actual data from AFM experiments. This study aims to address this gap by examining the effectiveness of a similar method. Such is done by training the models only on synthetically generated data and then analysing if they can accurately predict surface properties from AFM-studied samples.

1.2 Motivation and Objectives

Currently there is an urge to improve the postprocessing of data from AFM nanoindentations to infer the mechanical properties of a given sample, where each force-indentation curve needs to be individually fitted with an appropriate contact model, resulting in a time-consuming process.

Hence, the main objective of this work is to create an efficient framework that allows obtaining the surface properties of those samples, namely the Young's modulus and energy of adhesion, with a special focus on biological soft tissues, without relying on experimental data to build the ML models. To accomplish this goal, the following set of milestones was established:

- Understand the basic principles of Atomic Force Microscopy, its main operating modes and the most suitable contact mechanics theories to infer on mechanical properties from this kind of analysis;
- Explore the core concepts of Machine Learning and Artificial Neural Networks and their practical implementation in PyTorch;
- Generate synthetic data that could be representative of Atomic Force Microscopy Force-Indentation approach and retraction curves, for a low stiffness material;
- Develop, from scratch, two Deep Learning models capable of predicting surface properties from those approach (first model) and retraction (second model) curves, trained with synthetic data;
- Evaluate the first model based on the available experimental nanoindentation approach curves, that were fitted based on Hertzian theory;
- Generate three-dimensional surface maps and organize the experimental data within them, to later be predicted with the developed models, further extending the scope of this framework.

1.3 Thesis outline

This thesis is composed of a total 7 chapters, including this introduction, with Chapters 2 to 4 dedicated to conducting a comprehensive literature review on important topics that form the foundation of the framework.

Chapter 2 gives an overview of Atomic Force Microscopy, with a special focus on its working principle, operating modes and the necessary procedures to map mechanical properties from this analysis.

In Chapter 3, the Hertz and JKR contact mechanics theories are described and it is explained how each can be applied to measure the surface properties from samples that have undergone AFM nanoindentations.

Chapter 4 delves into Machine Learning, providing a concise view of its essential concepts and techniques. It focuses particularly on Artificial Neural Networks, ending with a demonstration on how to implement a Deep Learning framework in PyTorch.

The generation of synthetic data to train the proposed models is explained in Chapter 5, where the full creation, development and optimization of both Artificial Neural Networks models is traced step by step.

Taking advantage of experimental data from AFM nanoindentations, Chapter 6 presents the evaluation of the models built upon synthetic data, when facing the new dataset. In addition, the experimental data is rearranged in surface maps, which are then predicted by the models, also showing how this framework can aid in explicitly mapping surface properties.

At last, Chapter 7 summarises the main conclusions withdrawn from this thesis, highlighting some suggestions for further development of the framework presented.

Chapter 2

Atomic Force Microscopy

This chapter will focus on the key aspects of Atomic Force Microscopy. After a brief introduction, the working principle and operating modes of this method will be outlined, as well as required procedures to map the mechanical properties of each sample. Finally, an overview of its main applications is presented.

2.1 Introduction

In the last decades, many fields of science and technology have been turning their focus to increasingly smaller scales, as nanoscale systems pervade our everyday lives, being present in cellphones, computers, cars or medical equipment. Mechanical engineering and, in particular, the study of mechanical properties are following this trend toward nano/atomic scale, since material properties are fundamentally defined by the atomic structure.

In 1959, Richard P. Feynman hypothesized the possibility of reaching "nanotechnology" (without explicitly mentioning this word) down to the atomic scale (48). At the time, he predicted that to achieve this significant scientific progress, a series of decreasingly-sized machines would be required, each guiding the next smallest one. In the following years, it was discovered it was feasible to go down to the atomic scale in just one step from the macroscale, with further developments in electron microscopy.

The later invention of Scanning Probe Microscopy (SPM) allowed this technique to quickly establish as one of the most effective for nanoscale imaging. SPM provides resolution down to the atomic scale in real space, i.e., with instruments sized around 10 cm, images with a resolution of about 1 Å (10^{-10} m) can be accomplished (51). It uses a small probe to identify local properties of a surface, where a grid of points is scanned, so the detected properties can be mapped and expressed as an image. Generating an image of the sample topography is one of the most common applications of SPM, although images of many other properties can be obtained. Figure 2.1 shows the imaging range of SPM compared with other microscopy techniques.

Scanning Tunneling Microscopy (STM) was the first kind of SPM-based approach, invented in 1982 by Binnig and Rohrer (3), earning them the Nobel Prize in Physics four years later. In this

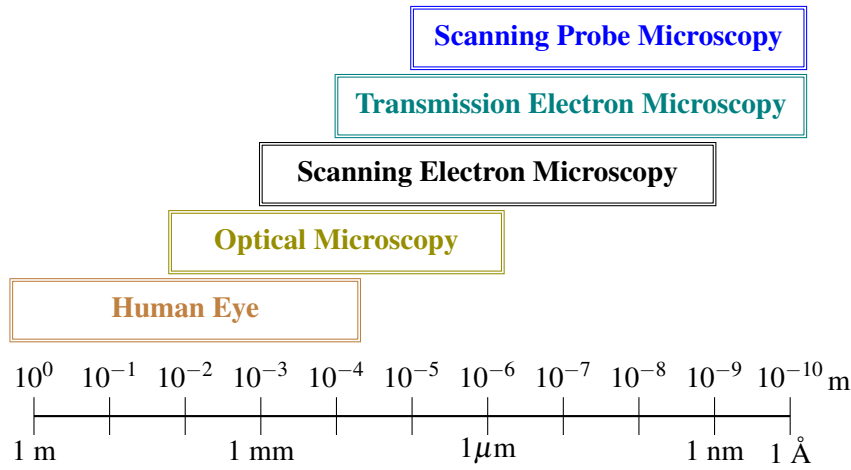


Figure 2.1: Comparison of imaging ranges for different microscopy techniques.

method, a voltage is applied between the tip of the probe and the sample, and a relation between tip-sample distance and the corresponding measured tunneling current is established. As the distance between tip and sample slowly decreases, the tunneling current increases exponentially, thus favouring a very precise control of tip-sample distance. A feedback mechanism is responsible for constantly adjusting the tip-sample distance, by retracting or approaching the sample or the tip, to a distance where the tunneling current is kept constant and equal to a preset threshold value.

One of the main disadvantages of STM is the fact that, since tunneling current is the quantity being measured, a conducting sample is required, which creates a major limitation for the application of this method, hence the urge for new techniques. Atomic Force Microscopy (AFM) - can also be referred to as Scanning Force Microscopy (SFM) - arose out of this need, allowing to measure the force between sample and tip, instead of the tunneling current, so insulating samples can also be tested. The Atomic Force Microscope was first described in 1986, by Binnig, Quate and Gerber (4), and besides paving the way for imaging and characterizing the mechanical properties of a wide range of materials, it had a major impact on the study of biological samples. In AFM, the sample surface is sensed by contact or near contact with a tip (that can have multiple shapes) placed on the end of a cantilever. The cantilever will act as a spring, with its deflection being proportional to the tip-sample force. So knowing the cantilever stiffness (spring constant, k), the force is determined by Hooke's law $F_{spring} = -kz_c$, where F_{spring} is the spring force and z_c is the distance the cantilever spring is bent, in relation to its equilibrium position (51). Working principle and main operating modes will be further explored in sections 2.2 and 2.3.

Figure 2.2 presents an example of an experimental AFM Force-Indentation curve, obtained from nanoindentations on smooth biological cells. As shown, the analysis can be decomposed in two stages: approach and withdraw, corresponding respectively to the moments when the tip moves towards the sample and then away from it, after reaching the maximum indentation value. The contact point matches a null indentation, so positive values in the x-axis represent contact between sample and tip, while the opposite happens for negative values, hence the more negative the indentation, the greater the distance between sample and tip. As for the force, one can note

that when the tip is far away from the surface, the interaction forces between them are negligible, despite being able to observe some fluctuations in this region. In this example, there are repulsive forces while tip and sample are in contact, and attractive adhesion forces while the tip is detaching from the cell. In non-contact mode, there are attractive (negative) forces when sample and tip are close to each other and repulsive (positive) forces when the distance between them gets even smaller.

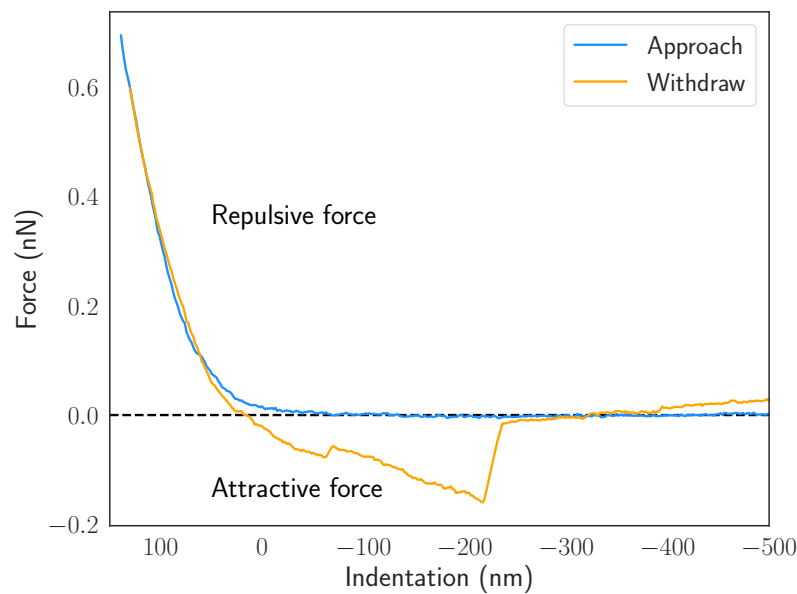


Figure 2.2: AFM Force-Indentation example curve, obtained for a smooth muscle cell.

2.2 Working Principle and Instrumentation

The most obvious potential of AFM was first identified in terms of imaging very small samples, since the surface structure can be obtained with remarkable resolution, allowing to get images displaying the distribution of individual atoms in a sample. In addition, it also stands out for its versatility, as almost any sample can be imaged, from hard ceramic materials, to flexible polymers or soft biological tissues. Other microscopy techniques are commonly based on directing light or electrons onto the surface. This forms a two-dimensional image, from which the height is not directly perceived, thus needing to be inferred from the image or requiring the sample to be rotated. In contrast, the sharp probe in AFM directly taps the sample, this way building a map of the sample's surface height. On the one hand, height information is obtained much more easily compared to other microscopy methods, but on the other hand it requires processing data from AFM to form a similar image to that of a microscope. Nevertheless, this data treatment is typically simple and once it is done, images capturing every perspective of the sample can be

easily generated, with suitable software, making it straightforward to measure each dimension of any feature in the image.

Besides its imaging capabilities, AFM has many other "spectroscopic" modes that measure sample properties at the nanoscale and which are more relevant to the main focus of this work. Nevertheless, whether the objective is to analyse the topography of the sample or to study its mechanical properties, there are some key features that can always be found in an Atomic Force Microscope: probe, optical detection system, piezoelectric scanner and feedback system. A simplified scheme of an AFM analysis is exhibited in Figure 2.3, where the interaction between laser, probe and photodetector is clear.

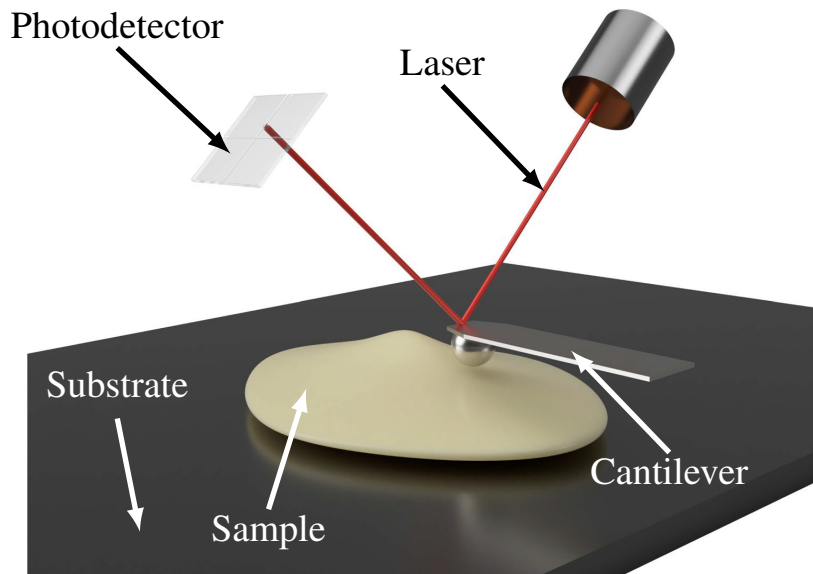


Figure 2.3: Schematic view of the AFM method (14).

The probe can be divided in two components: a tip and a cantilever. By directly interacting with the sample, the tip senses attractive or repulsive forces, depending on sample properties and surface topography. The mechanical properties of the probe are defined by the bending cantilever, which serves as its structural component and has the tip positioned at its end, as represented in Figure 2.4. There are two main methods for oscillating the cantilever: one involves using a piezoactuator that is in contact with a supporting chip, while the other consists in applying an alternating field to a magnetic film that is deposited on the back of the cantilever. Tip-sample interaction causes the cantilever to deflect, twist or change its harmonic oscillator characteristics. Thus, if the cantilever properties are known, the forces resulting from such interaction can be measured. Both tip and cantilever are usually made of silicon (Si) or silicon nitride (Si_3N_4). Cantilevers are generally rectangular or triangular, having a length of 100-200 μm , a width of 20-40 μm and a thickness of 0.5-1 μm . To enhance reflectivity, they are frequently coated with a layer of gold or aluminum. The most common tip shapes are tetrahedral, spherical or conical. Tetrahedral and conical tips frequently have a radius of 5-200 nm, while when it comes to spherical tips, the range is between 0.01-10 μm .

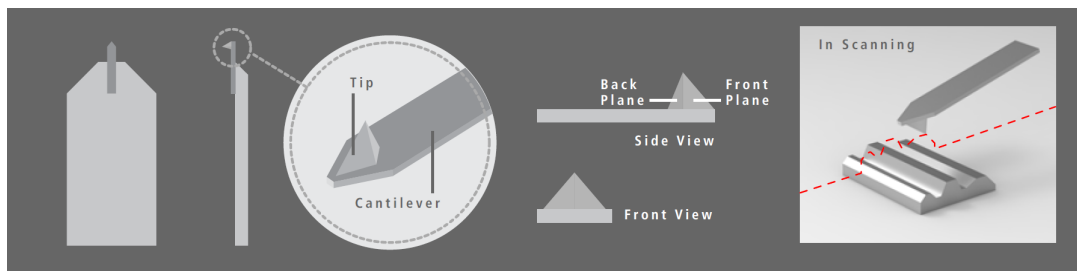


Figure 2.4: Views of a typical AFM probe (47).

Probe manufacturers always provide information about the properties of each model, such as the resonance frequency, spring constant and quality factor, in addition to the probe material and dimensions. Selecting an adequate probe is crucial to obtain good results from each specific type of AFM analysis. For instance, when it comes to imaging in non-contact mode, the probe must have high resonance frequency (>200 kHz) and high spring constant (>20 N/m), since under ambient conditions, a low spring constant cantilever often creates a contaminated layer on the sample, when its tip traps moisture. For contact mode imaging, a probe with low spring constant (<10 N/m) and low resonance frequency (<200 kHz) must be used, because a stiff cantilever can cause the tip to apply high forces to the surface, which can damage the tip itself, the sample surface or both. The shape of the sample must also be taken into account, as the acquired image results from the convolution of the tip shape with the sample surface. So, when there are details with the same length scale of the tip dimension, the convolution will have a negative impact in the imaging process, leading to a final image that doesn't accurately represent the real sample surface. To map nanomechanical properties, it is usual to work with a contact mode probe (with low frequency and low spring constant), but more accurate results are achieved when the probe selection is adapted to the sample properties: a soft cantilever is suitable for soft materials, while a stiff cantilever must be applied for hard materials (47).

The optical detection system is utilized for measuring cantilever deflection and torsional variations. It is composed of a laser diode, optical amplification components and a four-quadrant photodetector. At first, the laser beam is aimed at the cantilever backside, from which it is reflected to an optical amplifier and then collected at the photodetector. A deflection of 0.01 nm can be optically amplified 300 to 1000 times, so the photodetector detects a displacement of 3-10 nm, generating a measurable voltage that translates the force variation at the sample surface (10). Regarding the light collected in the photodetector, the output difference between upper and lower quadrants is proportional to the cantilever deflection, while the output difference in the lateral quadrants is proportional to cantilever torsion.

Another key part of an Atomic Force Microscope is the piezoelectric scanner, responsible for moving either the probe in relation to the sample, or the sample in relation to the probe. It is important to note that, as its name states, this scanner is based on the principle of piezoelectricity, which is a property that allows a material to deform when exposed to an electric field, or to generate an electrical charge in response to an applied mechanical stress or deformation. In AFM,

a piezoelectric material is deformed by applying an electric current, resulting in mechanical strain that is utilized for moving the probe or the sample. This allows very small distances to be scanned, related to the application of a calibrated amount of current. A greatly accurate control is ensured by the small expansion coefficient of the piezoelectric device, which is usually in the order of 0.1nm/V . There are three most common scanning possibilities: the first one relies in one scanner that moves the sample in x, y and z directions; the second uses one scanner to move the sample in x and y directions, and another to drive the probe in the z direction; while the third works with only one scanner, responsible for moving the tip in x, y and z directions (10).

At last, there is the feedback system, accountable for identifying height changes at the sample surface, based on cantilever deflection or oscillation, and then passing on information to the piezoelectric z -scanner or to the cantilever actuator, in order to minimize those variations. Sensitivity of the feedback system can be adjusted prior and even during AFM procedure.

To control all parameters in an AFM analysis, a software interface is needed. It allows moving the sample to the right location to be scanned, getting the probe near the surface, selecting the scan mode and scan parameters, besides displaying images while scanning or measuring Force-Distance (F-D) curves. These curves translate the force experienced by the probe as a function of distance from the sample surface. They are measured by moving the probe towards the sample until a pre-selected position, and then retracted, while the signal corresponding to the cantilever deflection is recorded throughout this movement. Several variables must be defined when getting a F-D curve, such as probe start and end positions, rate of probe motion, number and location of F-D curves to collect. This provides the opportunity to obtain several curves in a user defined grid pattern over the surface area, being essential to study the variation of the sample's mechanical properties along different locations of its surface.

2.3 Operating Modes

The availability of various AFM modes and experiments has made it a versatile and powerful tool. Initially, only contact mode imaging was available, limiting the types of experiments that could be performed and the type of data that could be produced. However, over time, there has been an explosion in the number of possible modes of operation of AFM, with at least 20 different modes now available. Many of these newer modes use the high resolution scanning of an AFM probe to measure different properties of the sample surface at the nanoscale. Overall, there are two main groups of AFM modes: topographic and non-topographic. The first group is related to measuring the sample topography, to later generate high resolution images of the sample. The second group is associated with the study of sample properties. Despite not being as common as the previously mentioned modes, surface modification is a third AFM operating mode, that relies on the accurate control of the probe motion over the sample, to manipulate the surfaces being analysed, based on local oxidation or scratching. Since the first two mentioned modes comprise the large majority of AFM applications, they will be the only ones to be further explored, and

considering that non-topographic modes have a higher relevance in the context of this framework, their description will be more detailed in comparison with topographic modes.

2.3.1 Topographic Modes

As stated in previous sections, the probe-sample interaction generates a map of height measurements, that can be subsequently transformed into a more realistic representation of the sample's shape, using light shading, perspective and other techniques. Topographic modes differ in the way height measurements are performed, since this can be done in contact mode, where the static deflection of the cantilever is measured, or dynamic mode, where the dynamic oscillation of this component is the measured quantity. Not only these differ in the experimental procedure, but also in the information that each presents.

Contact mode was the first developed for AFM and provided the basis for development of other more advanced modes. Despite being the oldest method, it still provides very high-resolution images, whilst being the fastest technique, since the cantilever deflection directly leads to the sample topography, without the need of summing oscillation measurements. In this mode, the tip is constantly in contact with the sample and the force is always repulsive. When cantilever deflection information is recorded to form the topographic image, the sample surface is scanned with the cantilever being kept at a constant height, allowing high scanning speed, but requiring smooth samples. There is another contact method, more widely used than the previous one, which consists in recording the cantilever height variations, while deflection is set at a constant value (hence having constant tip-sample force), permanently restored by a feedback loop as the probe goes through surface asperities. This allows for scanning hard samples, at the cost of the feedback loop lowering scanning speed. Since here the cantilever deflection doesn't change, the sample topography is acquired based on the amount of movement in the z direction that the piezoelectric device must undergo to keep the deflection constant. Plotting this information in relation to distance, gives the height image in contact mode.

Lateral force images can also be collected in contact mode, from lateral twisting of the cantilever, resulting from the topography and frictional forces exerted by the sample on the tip. The photodetector (normally made-up of four segments) determines vertical deflection by the difference in signal (in electric current or potential units) between the top two and bottom two segments, while lateral deflection corresponds to the signal difference between right and left segments. Figure 2.5 illustrates this in a schematic manner.

Despite allowing to obtain high-resolution images, contact modes have also some drawbacks. Due to the fact the tip and sample are always in touch, one of them (or even both) can be damaged during the scanning process and the presence of lateral forces can have a negative impact in the measurement of normal forces. Furthermore, the nature of the sample surface can also affect the accuracy of the data obtained, making it important to carefully consider the experimental conditions and potential sources of error.

Contrarily to contact modes, in dynamic modes the tip only contacts the sample intermittently, thus experiencing attractive and repulsive forces. Due to this reason, it is also called tapping

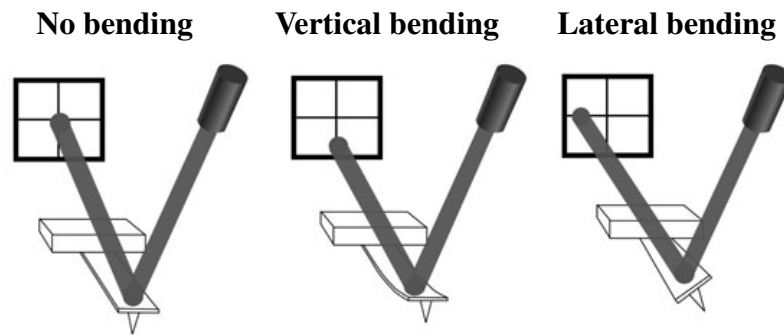


Figure 2.5: Photodetector identifying cantilever deflection and torsion. Adapted from (12).

mode. There is also the possibility for the tip to never be directly in touch with the sample, always oscillating above it and only experiencing attractive forces, in the so called true-noncontact mode. Dynamic (or oscillating) modes resort to an additional piezoelectric element to oscillate the cantilever, commonly with a frequency close to resonance and an amplitude of a few nanometers. As the tip is scanning the sample, parameters as cantilever height, amplitude and phase shift are monitored, and typically the amplitude is being kept constant by the feedback system to a preset value, although this monitoring can also be performed by fixing the other two parameters. Oscillating modes are particularly useful when imaging soft biological samples, as the almost negligible contact minimizes sample damage. Lateral shear forces that could induce error in contact mode are also eliminated. Additionally, it has been noted that accurate tip-sample control, which involves balancing the contribution of attractive and repulsive forces, will significantly improve lateral resolution in imaging applications (35).

Dynamic modes allow obtaining height, amplitude and phase shift images. Height images are collected by keeping the amplitude constant and recording the cantilever height, similarly to what happens in contact mode. Amplitude images show local amplitude changes, also exhibiting an edge accentuation effect that is independent of significant height changes, allowing for a clear observation of small details that otherwise could not be examined. Phase shift images illustrate the phase lag between the cantilever oscillation driving signal and its output signal. It provides information on mechanical properties related to elasticity, viscoelasticity, friction and adhesion.

2.3.2 Non-Topographic Modes

Non-topographic modes are associated with spectroscopic techniques, in a way they study sample properties instead of its topography. There are many non-topographic modes, such as force spectroscopy, nanoindentation, magnetic force microscopy, electric force microscopy, electrochemical AFM and thermal modes. Only the first two will be focused on, being the ones that rely on Force-Distance or Force-Indentation curves.

In force spectroscopy, the probe is moved along the z axis, while $x - y$ position is locked, and the cantilever deflection is reported as the probe approaches and retracts from the sample. High cantilever flexibility and great deflection sensitivity (provided by the optical detection system)

allow studying interactions between single molecules, and the tip is often modified with molecules of interest. Living cells, cell membranes and micro-organisms can be probed using this method, as well as more common materials, like polymers, metals and ceramics.

Figure 2.6 shows an example of what an ideal AFM Force/Deflection-Distance curve would look like. In the beginning of a force spectroscopy analysis, the probe is far from the sample surface, represented by point A. It then starts the approach phase and by getting closer to the sample, attractive forces will pull the probe towards it, creating the "snap-in" region. After establishing contact, as the probe advances, increasingly higher repulsive forces will be felt by the tip, until the preset value of maximum distance (or deflection) is reached in point B. After that, the probe starts withdrawing, until "pull-off" occurs at point C, where the force applied to the cantilever becomes higher than tip-sample adhesion. The "Adhesion data" region allows getting information on adhesion force and energy. The way the cantilever deflects when undergoing different stages of force spectroscopy is displayed in Figure 2.7, where each deflection (or resting) mode is associated to points A, B and C.

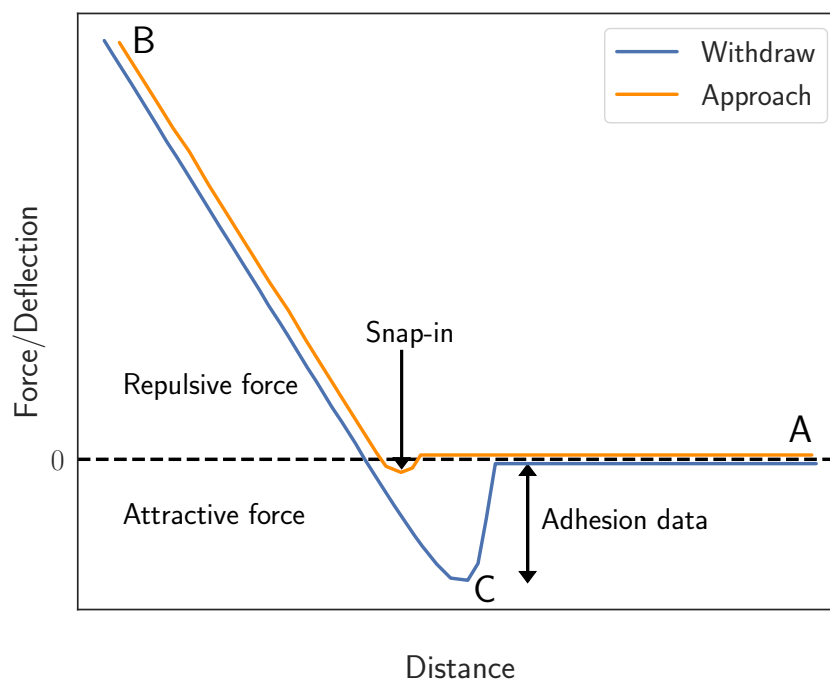


Figure 2.6: A model of an ideal AFM Force-Distance curve. A - Probe's initial position; B - Maximum indentation point; C - Point of maximum adhesion force. Adapted from (12).

However, experimental curves are not as clean as this ideal example, which can be observed by comparing it to the graph in Figure 2.2. Hence, to conduct force spectroscopy, various experimental factors need to be considered and addressed appropriately. For instance, the rate of approach and withdraw will affect the results and the tip radius can also have a great impact on the analysis,

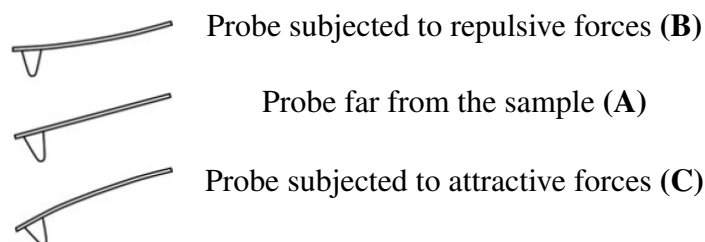


Figure 2.7: Illustration of the cantilever deflection in different AFM stages.

since it influences the number of molecules that will be in touch with the tip at once. AFM has the advantage of being performed in a wide variety of environments and for biological samples, it is common to do the experiment in a liquid solution (10). The composition of the liquid will then have a considerable effect in measured forces.

If we record the data while the tip of an AFM presses onto a sample surface instead of withdrawing from it, nanoindentation is being performed. However, this term has now been commonly used to describe the entire ramp process. Another method (unrelated to AFM) of nanoindentation exists, involving a specialized machine to measure load-displacement curves as a hard indenter presses into a sample. This machine, sensitive to forces at the micronewton level, can create a series of holes in a sample, and the size of these holes can be measured through techniques such as light microscopy. While performing a similar experiment, using AFM-based nanoindentation offers some advantages such as high load sensitivity (being able to reach the piconewton range), high positioning resolution (allows to probe very small regions of a sample), and the ability to measure indents at high resolution in x, y, and z directions. However, it also has some drawbacks, such as the probe approach not being perpendicular, non-linear z positioning, and the need for system calibration to extract real forces.

When hard materials undergo AFM nanoindentation, a very stiff cantilever and hard probe, such as a steel cantilever with a diamond tip, are necessary, although these hard probes produce lower-resolution images. Normal AFM probes can also be used for nanoindentation, but once again it is crucial to carefully characterize the tip radius and cantilever for quantitative results. This further characterization enables choosing probes with a broader spectrum of spring constants and prevents on solely relying on hard probes, that are not appropriate to study soft samples. One common approach to simplify tip radius determination is to use a normal AFM cantilever with a spherical tip. Data analysis for nanoindentation is often carried out through modeling the indentation using the Hertz contact model, which assumes only elastic compression and requires information on the tip shape and radius.

Both force spectroscopy and nanoindentation are often performed in a grid pattern over the sample, so the variation of measured mechanical properties along its surface can be determined and the softest or hardest regions are identified. Subsequently, heat map images of the sample surface can be generated for each measured property. It must always be taken into account that the sample topography can affect highly specific measurements, such as the tip-sample adhesion.

The way mechanical properties can be mapped based on Force-Distance curves, resulting from non-topographic modes will be further depicted in Section 2.4.

2.4 Mapping of Mechanical Properties

The most common measured properties based on AFM Force-Distance (F-D) or Force Indentation (F-I) curves are related to stiffness, hardness and adhesion. However, these curves are not directly generated in an AFM analysis, so there are a series of considerations that must be taken into account, to transform raw data from AFM, to the final F-D or F-I curves.

Throughout a non-topographic mode experiment, as the probe approaches and withdraws from the sample, the voltage measured in the photodetector is recorded as a function of the vertical displacement of the cantilever. In Figure 2.8, we can see that the photodetector measures the difference between the voltage corresponding to the laser reflection in the deflected cantilever and the reflection in the cantilever with its initial/undeformed slope (represented by the dashed laser reflecting on the dashed cantilever). As for the cantilever vertical displacement, it is expressed by the variable z_p , also referred to as the z-piezo displacement, that identifies scanner position. Hence, raw AFM output consists in voltage as a function of scanner position.

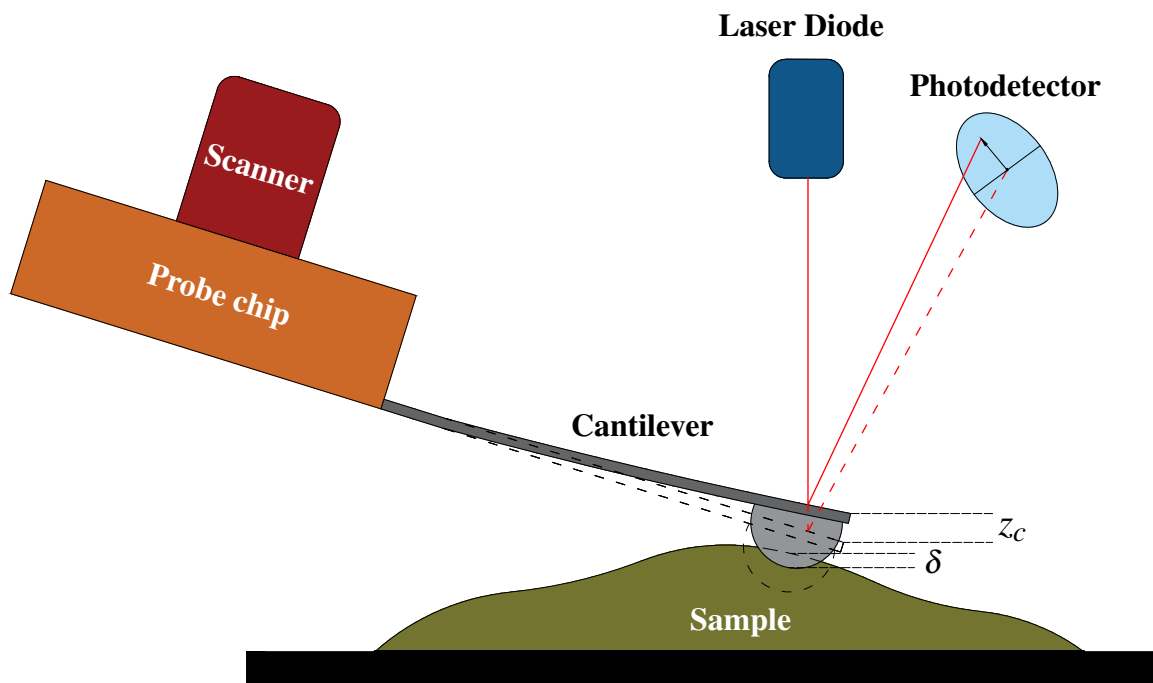


Figure 2.8: Experimental setup to obtain F-D or F-I AFM curves, where the dashed cantilever represents its original/undeformed shape. z_c represents the cantilever deflection and δ the indentation depth.

In addition, Figure 2.8 illustrates two key variables to AFM in general, with a significant relevance on processing the raw output data. One is the cantilever deflection, z_c and the other is the indentation depth, δ . Before performing force spectroscopy or nanoindentation, it is common to perform a full approach and withdraw cycle against an undeformable surface, to ensure that δ equals zero, and that during contact the z-piezo displacement is equivalent to the cantilever deflection. It also allows establishing the relation between the photodetector output (in volts) and the corresponding cantilever deflection, thus identifying the system sensitivity, commonly presented in V/nm . Figure 2.9 shows a typical curve from a pre-analysis with an undeformable sample, where the voltage output remains constant until there is contact, and then starts increasing, so that the slope observed in the graph is defined as the system sensitivity, S .

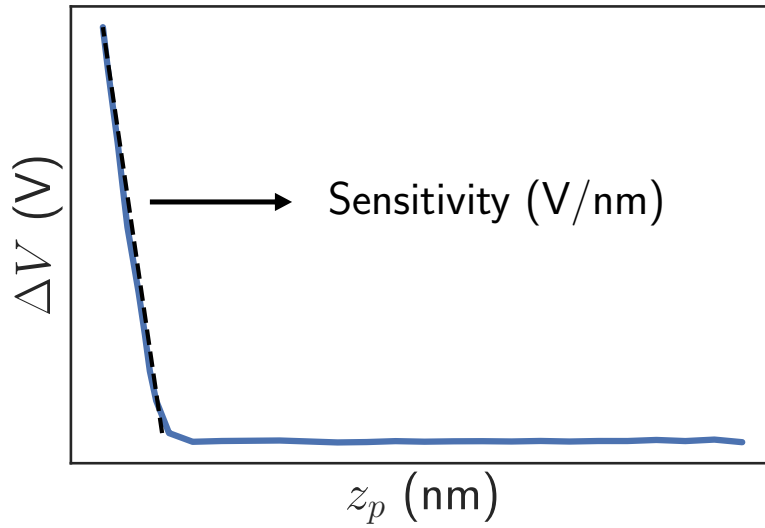


Figure 2.9: Example of the raw output data obtained from nanoindentation in an undeformable surface.

At this stage, it is already known that using expression 2.1, it is possible to transform the raw output (ΔV) into cantilever deflection (z_c .)

$$\Delta V = S \cdot z_c \quad (2.1)$$

When the analysis is performed in liquid media, smoothing algorithms must be applied to the curves, to correct any hydrodynamic dragging effects that may occur. Finally, the tip-sample force is calculated using Hooke's law (Equation 2.2), which requires knowing the cantilever stiffness (or spring constant, k). Indentation depth is determined using Equation 2.3 (only applied to the contact region), where z_0 is the zero indentation point, that can differ accordingly to the applied contact model. For instance, in the Hertz model it corresponds to the contact point. It is relevant to say that accurately establishing the contact point is one of the main challenges of AFM.

$$F_{spring} = -k \cdot z_c \quad (2.2)$$

$$z_p - z_0 = \delta + z_c \quad (2.3)$$

We then arrive at the final F-I curves, from which one can infer key mechanical properties, as the sample Young's modulus (E), work of adhesion (W_a) or adhesion force (F_{ad}). The first one is commonly measured with the approach curve, while the other two are obtained from the retraction curve. Likewise, the sample hardness H at the nanoscale can be obtained, only by dividing the maximum force experienced by the tip (F_{max}) and the projection of the contact area at maximum indentation (A_{max} , that can be calculated from empirical equations that rely on the tip shape): $H = F_{max}/A_{max}$.

Most properties are inferred using the right contact model for each AFM stage. The approach curve, where adhesion forces are not significant, is usually fitted with the Hertz model. As for the withdraw curve, where adhesive forces have a great impact in the experiment, Hertzian analysis is no longer valid, so other models must be employed. Amongst them, there are the JKR (Johnson-Kendall-Roberts), DMT (Derjaguin-Muller-Toporov) and Maugis models. JKR and DMT models represent two extreme cases in adhesive interaction: JKR is applicable for soft samples, small tip radius and high adhesion forces, while the opposite is verified for DMT, since it is valid for hard samples, large tip radius and low adhesion forces. The Maugis model represents a transition regime between the two previously mentioned extreme states. To determine the suitable model (that account for adhesion) for each case, the dimensionless Tabor parameter is calculated and it is given by:

$$\mu = \left(\frac{R\gamma^2}{E^2 s^3} \right)^{1/3} . \quad (2.4)$$

In Equation 2.4, R is the tip radius, γ is the energy of adhesion, E is the Young's modulus and s is the equilibrium separation between the surfaces. When $\mu < 0.1$ the DMT model must be applied, while for $\mu > 5$ the JKR model should be adopted. Typical experimental setup for soft biological samples or gels will present a Tabor parameter value of $\mu \approx 200$, hence the JKR model frequently being the preferred choice (13).

Both JKR and Hertz models assume the sample as being purely elastic and since these are the most relevant models to this framework, they will be properly presented in Chapter 3.

Despite all the advantages inherent to mapping the mechanical properties based on AFM F-I or F-D curves, it still has some limitations. The main limitation is determining the zero indentation or contact point, which is key to reach the final F-I data, specially when there is no snap-in region in the curve. Furthermore, for soft samples it might be hard to distinguish between surface forces and sample deformation. Some problems can also arise from hysteresis and creep in the piezoelectric scanner, that can lead approach and withdraw curves to overlap or swap positions if the load rate is too small and the displacement is too large. In addition, postprocessing requires a significant

amount of time, since manual observation is required to assess curve quality. Lastly, given the high number of points in each curve and the typical need to analyse thousands of curves, the process of selecting a contact model and applying it to fit the data can be a challenging and time-consuming task. Hence, alternatives to this procedure are being explored, as the one presented in this work, based on Machine Learning models. These models possess tremendous potential as robust tools in AFM postprocessing, that could contribute to enabling the precise determination of contact and detachment points, identifying low-quality curves, and extracting valuable insights regarding mechanical properties and other relevant analyses.

2.5 Applications

In previous sections, the versatility of AFM has been highlighted as to its ability to operate in a great number of modes and to study very different materials. This section aims to present some works on those diverse applications, to get a better understanding of what is currently being done in the scope of AFM and what are the future developments that this technology will promote.

It has become common to use AFM to study adhesives and nanocomposites, as described in (33), where a morphological analysis of an epoxy resin, filled with graphene-based nanoparticles is complemented with a rheological behaviour study. In dynamic mode experiments, the AFM morphology analysis allowed observing the distribution of the nanofillers in the epoxy matrix and confirm that it was compliant with rheological and viscoelastic properties of the resins. For each measured property, a map was created, characterizing its distribution over the sample surface, to facilitate the association of each morphological detail with its corresponding properties.

When it comes to the study of polymer nanocomposites, AFM is a great tool, as it doesn't require any special treatment that would damage the tip nor the sample. Whether in contact or dynamic modes, it has clear advantages over other common methods: can be performed in ambient air (unlike electron microscopy, which requires vacuum) and offers a better resolution than Scanning Electron Microscopy (AFM provides a true 3-D surface profile, whereas SEM only produces 2-D projections), as stated in (44).

The field of tribology also benefits from AFM, in the endless search for gear wear reduction and efficiency increase. Thin film analysis has been successfully performed with AFM, by mapping the peak force over the sample surface. For instance, the study conducted in (42) describes the mechanical characterization of ashless tribofilms, using nanoindentation, lateral force microscopy and peak force mapping.

Bioengineering and biomechanics often recur to AFM analysis, due to its efficiency in dealing with soft samples. It has become common to try and simulate the behaviour of a biological sample undergoing AFM indentation, using Finite Element Method (FEM) software. In (14), both experimental and simulation (based on the FEM) setups were produced, to analyse mechanics and dynamics of a cell cortex during indentation. Viscoelasticity of the cell was also studied, by keeping the probe in place at the maximum indentation depth. In the biomechanics field, it is also usual to produce surface maps of the measured properties, as performed in (5), where the elastic

modulus of the cortical bone is studied through AFM nanoindentations, with the resulting stiffness map being illustrated in Figure 2.10.

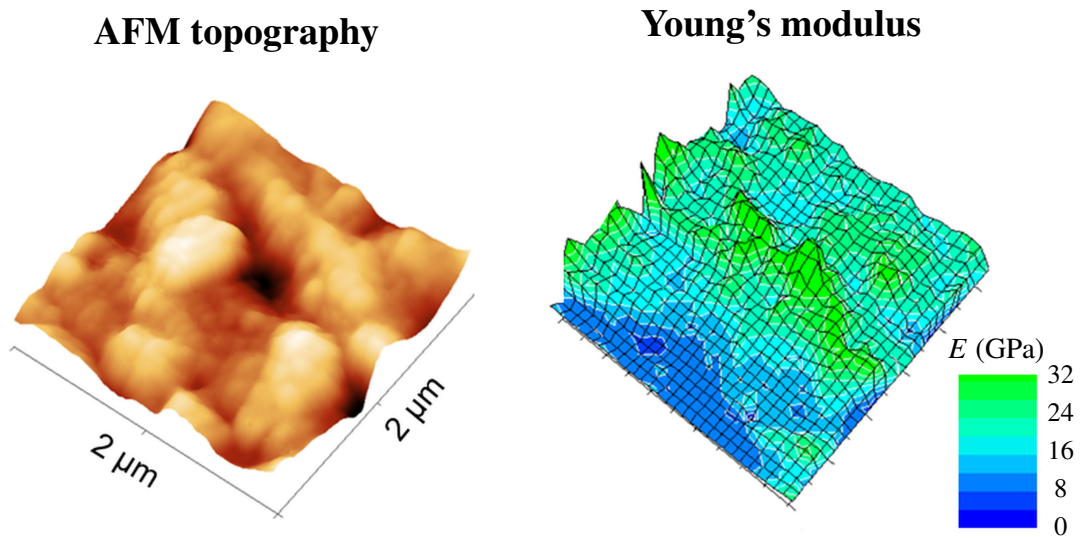


Figure 2.10: Young's modulus map from a cortical bone sample, obtained with AFM nanoindentations (5).

To overcome limitations in AFM data processing, new research has been produced to replace regular contact model fitting with Machine Learning models. It is the example of (30), where Gaussian Process Regression, Multiple Linear Regression, Random Forest and Support Vector Regression are applied to determine the elastic modulus of the sample, without requiring any contact model fitting. ML models presented accurate results, motivating the further development of similar approaches.

Chapter 3

Contact Mechanics

Contact mechanics will be the focus of this chapter. At first, a short introduction on the topic will be presented, followed by a more in-depth analysis to the two contact models that are more meaningful for this work, based on Hertz and JKR theories.

3.1 Introduction

Contact mechanics, as traditionally understood, refers to the way in which solids behave when they come into contact with one another and experience an external force. The origins of contact mechanics can be traced back to the work of Heinrich Hertz in 1882 (16). Hertz's original theory of contact mechanics was limited to the study of perfectly elastic solids or surfaces without friction or adhesion. Later, more sophisticated theories to describe the behavior of viscoelastic solids in contact were developed. However, none of these models took into account the role of interfacial adhesive interactions between the contacting surfaces.

In 1971, Johnson, Kendall and Roberts (17), observed that small particles undergo more deformation than predicted by the Hertz theory when they make contact with a flat surface or each other. Johnson's team attributed this excess deformation to attractive forces between the contacting surfaces, and they modified the Hertz theory accordingly. The resulting theory, known as the JKR theory, has since become widely used in the field of contact mechanics to study the adhesion and friction of various materials. Over the last few decades, contact mechanics principles and the JKR theory have been employed extensively to investigate these phenomena.

In recent years, with the advancement of computational methods, contact mechanics has been studied extensively using various computational tools. The main approach is based on FEM, which allows the simulation of complex contact problems using numerical methods. It has been employed to study contact between materials with different properties, such as elastic, viscoelastic, and plastic materials, as well as rough surfaces. Overall, the application of different computational methods has greatly expanded the scope of contact mechanics research, allowing for the investigation of a wider range of materials and geometries, as well as the study of contact mechanics at different scales, from the atomic to the macroscopic level.

3.2 Hertz Theory

3.2.1 Basic Principles

Upon bringing two elastic bodies into contact and subsequently applying a load, a localized deformation will occur in the contact region where they meet. What initially is a contact point will then increase to a contact surface. If both bodies are spheres, as in the case that will be used to illustrate this contact model, the area of the contact surface corresponds to a circle. By knowing the geometry and elastic properties of both bodies, conventional Hertz theory allows determining solutions for contact area, deformation, pressure distribution and stress at the contact area.

Hertz theory can only be applied to problems that satisfy the following assumptions: the bodies in contact only undergo elastic deformation, the surfaces are continuous and frictionless, and the dimensions of the contact area are small in relation to the curvature radius and to the dimensions of the bodies. Hence, despite the radii varying as deformation occurs, they are considered to be a constant over the very small regions near the contact area (54).

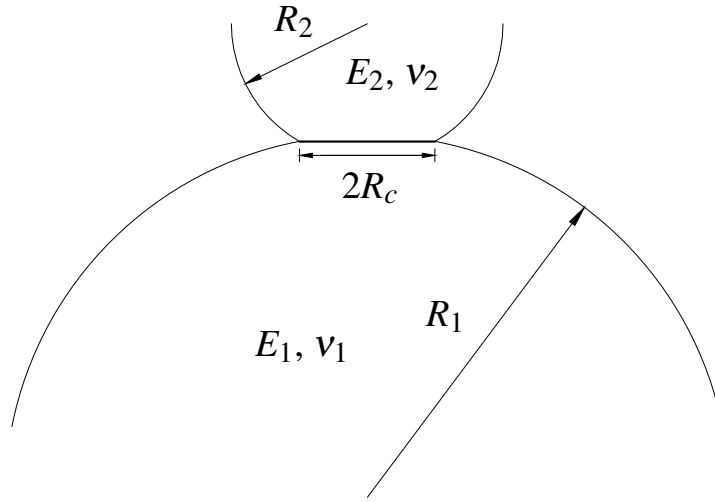


Figure 3.1: Scheme of two spheres in contact.

Starting with the example in Figure 3.1, we have two spheres in contact, with different radii (R_1 and R_2) and different material properties, where E_1 and E_2 are the Young's moduli, while ν_1 and ν_2 are the Poisson ratios. The radius of contact R_c is amplified only for illustration purposes.

An effective Young's modulus E^* is defined, based on Poisson ratios and Young's moduli of the two materials in contact, through the following relation:

$$E^* = \left[\frac{(1 - \nu_1^2)}{E_1} + \frac{(1 - \nu_2^2)}{E_2} \right]^{-1} \quad (3.1)$$

This equation translates the ability of the system to deform at a fixed contact area. It is also usual to define a radius R that accounts for both radii of curvature, R_1 and R_2 :

$$R = \left[\frac{1}{R_1} + \frac{1}{R_2} \right]^{-1} \quad (3.2)$$

At this point, the relation between the contact radius R_c and the applied force F , based on Hertz theory, can be introduced:

$$R_c = \left(\frac{3RF}{4E^*} \right)^{1/3}. \quad (3.3)$$

The contact displacement δ can also be defined as a function of the curvature radii and the distance between the centers of the spheres d :

$$\delta = R_1 + R_2 - d. \quad (3.4)$$

It can also be written as:

$$\delta = \frac{R_c^2}{R}, \quad (3.5)$$

which allows defining the contact area A in relation to R and δ :

$$A = \pi R_c^2 = \pi R \delta. \quad (3.6)$$

Combining Equations 3.3 and 3.5, we arrive at the force predicted by Hertz theory, when knowing the elastic properties of the materials (E^*), the radii of curvature (R) and the contact displacement δ (which corresponds to the indentation depth when we are referring to an AFM analysis):

$$F = \frac{4}{3} E^* R^{\frac{1}{2}} \delta^{\frac{3}{2}}. \quad (3.7)$$

It is often key to know the normal pressure distribution in the area of contact, as a function of the distance from the center of this region, and it can be calculated by:

$$p(r) = p_0 \left(1 - \frac{r^2}{R_c^2} \right)^{\frac{1}{2}}, \quad (3.8)$$

where p_0 stands for the maximum contact pressure, that is verified in the center of the contact circle. It is possible to determine this variable through the following expression:

$$p_0 = \frac{1}{\pi} \left(\frac{6FE^{*2}}{R^2} \right)^{\frac{1}{3}}. \quad (3.9)$$

The principles of Hertzian contact mechanics were originally formulated to describe the interaction between two elastic spheres, and despite the formulas presented being for this specific case, for some different contexts only slight modifications would be required. For instance, if we

had the common case of one sphere ($R_1 = R_{sphere}$) in contact with an elastic half space ($R_2 \rightarrow \infty$), Equation 3.2 would simplify to $R = R_{sphere}$. It implies that for a sphere and elastic half space contact scenario, Equation 3.7 would still be appropriate to calculate the predicted force. It is worth mentioning that an elastic half space is an isotropic and homogeneous material, with an infinite extension in all directions (including depth), only having the surface as a boundary.

Another common situation is having one body with a much higher stiffness than the other ($E_1 \ll E_2$), as it happens when performing AFM nanoindentation to soft biological samples, since the indenter is infinitely more rigid than the sample. Here, the contribution of the term related to the rigid body in Equation 3.1 will be negligible in comparison with the other one, so usually it is not considered and the effective Young's modulus is given by $E^* = \frac{E_1}{1-\nu_1^2}$.

When it comes to the application of this contact model to fit AFM data, some refinement was required, so the Hertzian theory has been expanded by Sneddon (45), deriving the relation between indentation depth and applied force for axisymmetric indenter shapes. For paraboloid shaped indenters, a similar approach to Equation 3.7 is suitable, while for conical indenters, the following expression must be applied (23):

$$F = \frac{2}{\pi} E^* \tan \alpha \delta^2, \quad (3.10)$$

where α is the cone half angle. However, equations deduced by Sneddon require the AFM tip to have simple shapes (conical or paraboloid), which is often not the case, so further refinement of these equations must be performed to allow analysing AFM curves in a more exact manner.

3.2.2 Overcoming Hertzian Theory Limitations in AFM Analysis

Many works have been developed to achieve an accurate relation between applied force and indentation depth, for a wider range of AFM tips, constantly using Hertz principles as a foundation. The equations that will be presented below refer to the case of contact between an indenter and an elastic half space, so the radius of curvature R will always correspond to the tip radius. In addition, the indenters will be considered as rigid bodies when compared to the sample (with $E_{sample} = E$), so $E^* = \frac{E}{1-\nu^2}$. Briscoe et al. (7) came up with the expression for pyramidal indenters, that usually have a round tip apex, approximating them to have a sphero-conical shape. This expression can be written as:

$$F = \frac{2E}{1-\nu^2} \left\{ R_c \delta - \frac{R_c^2}{2 \tan \alpha} \left[\frac{\pi}{2} - \arcsin \left(\frac{R_t}{R_c} \right) \right] - \frac{R_c^3}{3R} + (R_c^2 - R_t^2)^{\frac{1}{2}} \left(\frac{R_t}{2 \tan \alpha} + \frac{R_c^2 - R_t^2}{3R} \right) \right\}, \quad (3.11)$$

where R_t is the transition radius between the spherical and conical part of the indenter, that can be calculated by $R_t = R \cos \alpha$ if the spherical tip merges smoothly with the body of the cone. As for the indentation depth with pyramidal indenters, it is given by:

$$\delta = \frac{R_c}{\tan \alpha} \left[\frac{\pi}{2} - \arcsin \left(\frac{R_t}{R_c} \right) \right] - \frac{R_c}{R} \left[(R_c^2 - R_t^2)^{\frac{1}{2}} - R_c \right]. \quad (3.12)$$

For spherical indenters, Sneddon (45) has also proposed the following expression:

$$F = \frac{E}{2(1-\nu^2)} \left[(R_c^2 + R^2) \ln \left(\frac{R+R_c}{R-R_c} \right) - 2R_c R \right]. \quad (3.13)$$

It relates the contact radius to the indentation depth at a specific moment of the analysis, through:

$$\ln \left(\frac{R+R_c}{R-R_c} \right) = \frac{2\delta}{R_c}. \quad (3.14)$$

Despite giving an accurate prediction of applied force when spherical indenters are used, Equation 3.14 doesn't provide a direct relation between force and indentation depth, as there is a relation between R_c and δ at every position of the sphere, $R_c = R_c(\delta)$. Hence, it is not so practical to apply it in AFM nanoindentation. An alternative approach, based on applying a correction factor Z to Equation 3.7, has been proposed in (21):

$$F = \frac{4}{3} \frac{E}{(1-\nu^2)} R^{\frac{1}{2}} \delta^{\frac{3}{2}} Z. \quad (3.15)$$

In (20), the following expression has been proposed to compute Z :

$$Z = c_1 + \sum_{i=2}^N \frac{3}{2i} c_i R^{(\frac{3}{2}-i)} \delta^{(i-\frac{3}{2})}, \quad (3.16)$$

where constants c_i depend on the ratio between indentation depth and tip radius. A simpler approximation has been presented in (27):

$$Z = \left[1 - \frac{1}{10} \frac{\delta}{R} - \frac{1}{840} \left(\frac{\delta}{R} \right)^2 + \frac{11}{15120} \left(\frac{\delta}{R} \right)^3 + \frac{1357}{6652800} \left(\frac{\delta}{R} \right)^4 \right]. \quad (3.17)$$

The relation between force and indentation depth (with only positive values in the x -axis) computed from Equation 3.15, approximating Z by Equation 3.17, is shown in Figure 3.2, for different Young's modulus values, while the other variables remain constant.

Finally, several methods for axisymmetric indenters with arbitrary shapes have been suggested. Most rely on the concept of "effective indenter", which takes its geometry from the shape of the plastic hardness impression formed during loading (36). It can be determined as a function of a fitting constant B , a variable depending on the material properties d and the radial distance from the center of contact r :

$$z = Br^d. \quad (3.18)$$

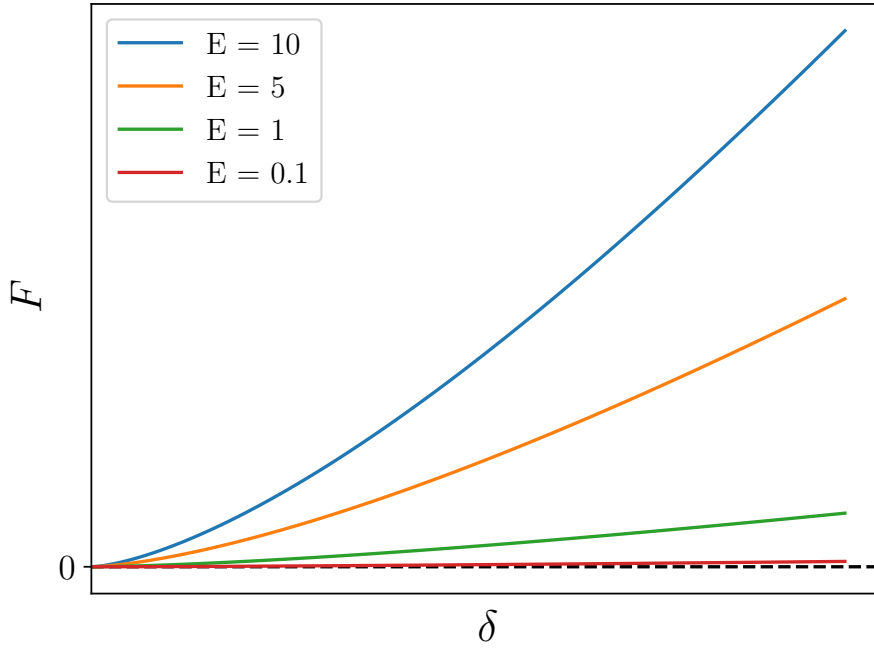


Figure 3.2: Relation between force and indentation depth, computed through Hertz theory- based approximation in Equation 3.15.

A general formula for axisymmetric indenters has been given in (22) and can be written as:

$$F = \frac{2d}{d+1} \frac{E}{(1-\nu^2)} R_c \delta, \quad (3.19)$$

which still has the disadvantage of requiring the calculation of the contact radius for each indenter shape, or to define it as a function of indentation depth.

3.3 JKR theory

By studying the influence of adhesion energy (interchangeably referred to as surface energy) on the contact between elastic solids, Johnson, Kendall and Roberts (17) derived valuable equations translating the effect of surface energy upon adhesion force and contact size. They started by analysing the contact between two spherical solid surfaces (as Hertz had done), using rubber and gelatine as materials.

Surface energy arises due to cohesive forces between the atoms or molecules within a material. It is related to the work that is needed to separate bodies in close contact and to overcome the adhesive forces between them, being expressed in energy per area units.

The need for a new contact model that would replace Hertz theory in some cases, emerged from observations that at low loads, the contact areas between the bodies would be much larger

than what was predicted by Hertz, because strong adhesion was observed between them. However, at higher loads, Hertz theory presented good results. Hence, attractive forces were being observed between the bodies, and despite being almost negligible at high loads, they had a great impact as the load was getting closer to zero.

Figure 3.3 illustrates the difference of what happens when there are surface forces in play (generating contact radius R_{c1}) and in the absence of these forces (contact radius R_{c0}), besides showing the "new" surfaces that are created when there is adhesive contact. This last case is accurately modelled with Hertzian theory and R_{c0} can still be determined through Equation 3.3.

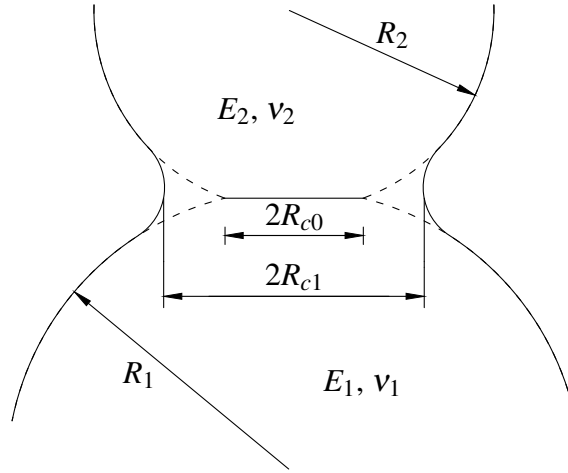


Figure 3.3: Scheme of two spheres in contact, with (R_{c1}) and without (R_{c0}) surface forces.

However, to infer R_{c1} (which from now on will be referred to only as R_c), the impact of adhesion energy must be accounted for, given that it will cause tensile stresses between the surfaces at the edge of the contact area and compressive stresses will only be present in the centre. Johnson, Kendall and Roberts then modified Hertz contact radius equations, into:

$$R_c^3 = \frac{R}{K} \left(F + 3\gamma\pi R + \sqrt{6\gamma\pi R F + (3\gamma\pi R)^2} \right), \quad (3.20)$$

in which γ is the energy of adhesion (in units of energy per area) and K relates to the effective Young's modulus by $K = \frac{4}{3}E^*$. We can see that when $\gamma = 0$, it goes back to Hertz equation $R_c^3 = \frac{3RF}{4E^*}$. At zero applied load, we get:

$$R_c^3 = \frac{6\gamma\pi R^2}{K}. \quad (3.21)$$

When the applied load becomes negative, the bodies are moving away from each other and the contact radius will decrease. For Equation 3.20 to remain valid, the following condition must be verified:

$$6\gamma\pi R F \leq (3\gamma\pi R)^2, \quad (3.22)$$

leading to

$$F \geq -\frac{3}{2}\gamma\pi R. \quad (3.23)$$

This allows calculating the force of adhesion F_{ad} at which the separation of the spheres will occur, being defined by:

$$F_{ad} = -\frac{3}{2}\gamma\pi R. \quad (3.24)$$

It is thus verified that this force is independent from Young's modulus, so this property will not interfere with the instant at which the bodies break adhesive contact. As for contact displacement δ , it is established by JKR theory as:

$$\delta = \frac{R_c^2}{R} - \frac{4}{3}\sqrt{\frac{R_c F_{ad}}{RK}} \quad (3.25)$$

At last, to fully describe an elastic model including the effects of surface energy, the total normal force of contact with cohesion F can be written as:

$$F = \frac{R_c^3 K}{R} - \sqrt{6\pi K \gamma R_c^3} \quad (3.26)$$

Overall, the JKR contact model provides an accurate approximation for large cohesive energies and low Young's modulus, making it well-suited to fit data from biological samples in AFM studies. Nevertheless, the application of this theory is not very straightforward, since the contact radius is a function of indentation depth, so there is not an explicit expression to relate this last parameter with the force. As done in (11), one way to tackle this problem is by approximating the contact radius simply as:

$$R_c^2 \approx R\delta. \quad (3.27)$$

Replacing this in Equation 3.26, leads to an easier way to calculate the normal force:

$$F = KR^{\frac{1}{2}}\delta^{\frac{3}{2}} - U_a K^{\frac{1}{2}} R^{\frac{3}{4}} \delta^{\frac{3}{4}}, \quad (3.28)$$

and U_a relates to the adhesion energy by

$$U_a = \sqrt{6\pi\gamma}. \quad (3.29)$$

It is clear that this approximate way to calculate the contact radius will slightly decrease the accuracy of the model, but by giving an explicit expression of the force as a function of the overlap, the complexity of the model decreases significantly, so does the analysis of data from AFM nanoindentation.

Illustrating the correlation established in Equation 3.28 is also of key importance. Unlike Hertzian theory equations, now there are two material properties ($K = K(E)$ and γ) affecting the

interdependence between force and indentation. If the adhesion energy was set as a constant and only the Young's modulus was changed, the relation between the different curves would be similar to what was presented in Figure 3.2, regarding Hertz theory. However, by adjusting the energy of adhesion and fixing the remaining variables, the plots in Figure 3.4 are obtained. It is noticeable that by increasing the surface energy, there are increasingly negative forces, meaning that adhesion forces become more relevant.

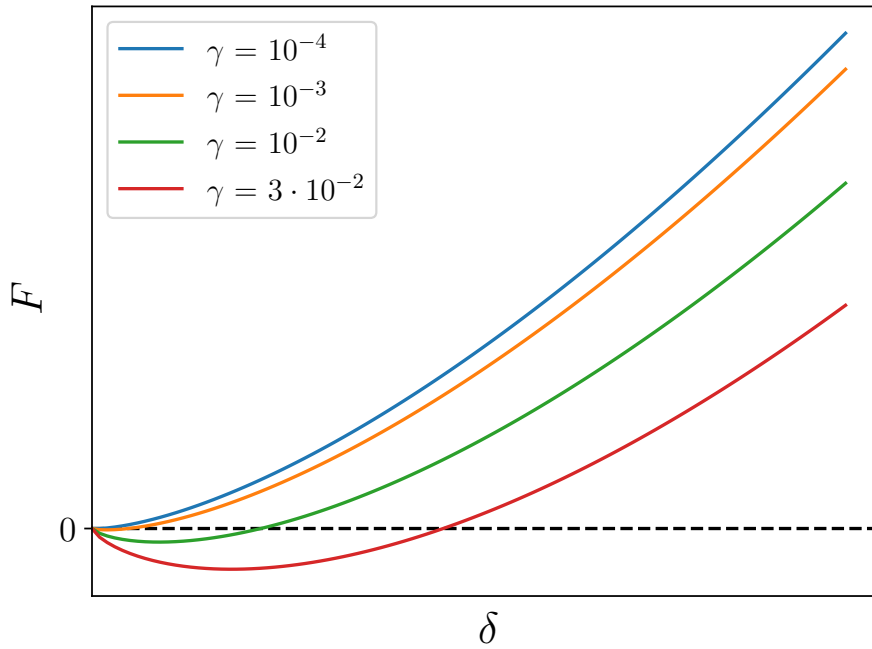


Figure 3.4: Relation between force and indentation depth, computed through JKR theory-based approximation in Equation 3.28, setting the Young's modulus as an unitary constant.

Chapter 4

Machine Learning

The goal of this chapter is to outline the main aspects related to Machine Learning, with an increased focus on the key features for the development of this work. The topic will be introduced and after that, Artificial Neural Networks will be described, as well as all the steps required in a Train-Validation-Test framework. The theoretical review presented along this chapter is mostly based on the books by Nielsen (32) and Géron (15).

4.1 Introduction

Artificial Intelligence (AI) and Machine Learning (ML) are two different concepts, although related, whose definitions are regularly mixed up. Artificial Intelligence only refers to the capability of machines to mimic human intelligence. On the other hand, Machine Learning is a branch of AI (Figure 4.1) associated with the development of algorithms capable of data-driven decisions, without the need to follow static programming instructions. It is said that based on a given dataset, the algorithm is trained to perform a certain task, by identifying patterns in the data, from which it will relate the specified inputs to the desired outputs. Another concept that is worth describing is Deep Learning (DL), a subset of ML referring to techniques that teach computers how to learn by example, using labeled datasets and neural network (NN) architectures.

ML has never been as trending as it is nowadays and its importance in today's society only tends to increase. Despite that, its foundations started being developed in the middle of the last century. Walter Pitts and Warren McCulloch, in 1943, presented for the first time a mathematical model of a neuron and described how networks of these artificial neurons could perform logical operations (25). In 1950, Alan Turing introduced the concept of a machine demonstrating intelligent behaviour if it would be able to pass the Turing test (49). Intelligent machines started to receive widespread attention when the first checkers playing AI was conceived by Arthur Samuel, and even more recognition as millions watched IBM's Deep Blue defeat the long-time world chess champion Garry Kasparov, in 1997.

Over the years, with exponential improvement in computing technology, development of massive datasets and advancement in computer science and statistics, ML models have been able to

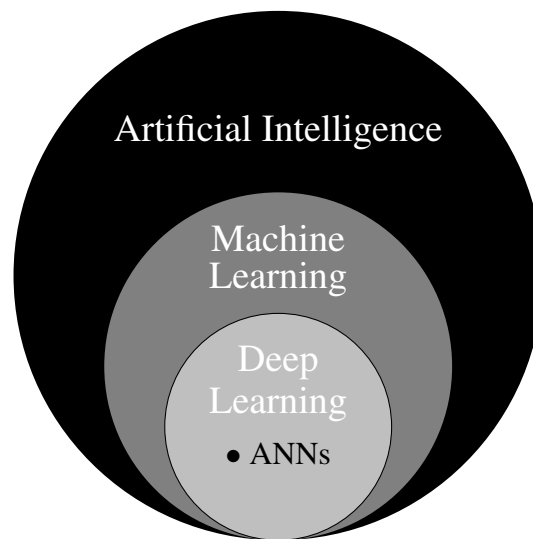


Figure 4.1: Illustration of the most important AI subsets for this framework, where ANNs stands for Artificial Neural Networks.

become a key part of our everyday lives. The first example having an impact of hundreds of millions of people was the spam filter in the 1990s, and since then it can be found in recommendation systems of every streaming service, used as a cancer prediction tool and being the key feature that allows developing self-driving vehicles.

From the distinct approaches to train models and extract patterns from a dataset, one can subdivide ML algorithms in two groups: supervised learning and unsupervised learning.

Supervised learning is the most common type of Machine Learning application and it is characterized by using labeled datasets, designed to train the algorithms into classifying data or predicting outcomes. The accuracy of the model is calculated by relating the true labels with the outputs of the ML model. Classification and regression are the two types of supervised learning problems. Classification problems correspond to separating data into different categories and the most common classification algorithms are linear classifiers, support vector machines, random forest and decision trees. Regression problems resort to algorithms that find the relation between dependent (outputs) and independent (inputs or features) variables, usually to predict numerical values. Linear regression, logistic regression and polynomial regression are the most popular algorithms belonging to this category.

Unsupervised learning, as the name suggests, takes unlabeled datasets and discovers hidden patterns within it, without requiring any human interference. Clustering, which is a technique to group unlabeled data based on their similarity, is the most frequent application of unsupervised learning and K-means clustering is the most common of its algorithms. Other tasks performed by unsupervised learning algorithms are association, that finds relations between the variables of a dataset (used for recommendation engines) and dimensionality reduction, that is used to decrease the number of features when it is too high, while maintaining the most relevant information in the dataset.

There are many ML frameworks built in numerous programming languages, but currently one particularly stands out as being preferred for research in this field and that is PyTorch. PyTorch (34) is an evolution of the open-source library Torch, written in Python, and all ML implementations in this work were made using this framework. To illustrate its relevance, in 2022, over 60% of (open-access) published papers related to ML were performed using PyTorch (53).

4.2 Train-Validation-Test Framework

To build a ML model and improve its performance based on a given dataset, a standardized approach has been established, based on splitting the dataset into three different groups: the training, validation and testing sets. Separating the data in this way ensures that the model is trained and evaluated in a rigorous and unbiased manner, and that the right hyperparameters are chosen by the developer.

First, it is key to define hyperparameters and to distinguish them from model parameters. So, a hyperparameter is adopted prior to training the model or evaluating its behaviour and it isn't learned from the data. A model's performance is highly dependant on hyperparameters, so it is usual to select the optimal values for them, in a separate procedure from training the model. The number of hidden layer and number of nodes in each layer are hyperparameter examples, to which we can add the learning rate or the activation function, that will be explored in following sections, amongst others.

On the other hand, the parameters of a ML model are internal variables, that are not directly set by the developer. They are responsible for capturing the patterns in training data and to produce predictions that are ever closer to the actual outputs, being improved through the application of optimization algorithms. The existing parameters in a ML model depend on its nature and architecture, but when it comes to ANNs the model parameters correspond to the weights and bias for each neuron in the network.

Now moving on to the dataset split, it should be noted that the biggest portion of the entire data goes to the training set. That is because the training process has the purpose of optimizing the model parameters, to increase its accuracy. Hence, a large fraction of the data must be provided so that the relationships between the different features and the corresponding outputs are correctly identified and modelled. In ANNs, the training process is strongly linked with the backpropagation algorithm (to be explored in section 4.3.4).

The validation set is much smaller than the training one, but plays a significant role when building a ML framework. This set is not used in training, thus when applying the model to the validation set, a good comprehension on how it behaves on "unseen" data is obtained, from which the hyperparameters are fine tuned. It is key to prevent overfitting to the training set, that translates the tendency that a model has to "memorize" the data instead of "understanding" its patterns.

At last there is the test set, that is usually similar to the validation set in terms of size and it is also disregarded during training. It ultimately defines the performance of the model.

Some concerns must be taken into account while performing the training-test-validation split. The first and most obvious one is to choose appropriate split ratios. The second is to apply a splitting technique called stratification, which is defined as ensuring that the output representation across the three sets is proportional to their original distribution. For instance, it prevents a model that deals with continuous numerical values from training with a set that predominantly contains lower values of the initial output range and then test it in a set with the higher values of that range, resulting in a poor model evaluation.

To make the training process more computationally efficient, instead of updating the parameters only after processing the entire training dataset, the parameters are updated after feeding the model with a subset of training data, called a batch (or mini-batch). The combination of all batches corresponds to the training set. This way, for one iteration through the entire set, the model is updated a number of times equal to the number of batches, rather than being updated just once.

After the model has seen all batches and therefore the entire training set, one epoch is completed. An epoch is a single pass through the whole training data, where each training instance has been processed and used to update the model's parameters. The number of epochs determines how many times the model will iterate over the entire dataset during training.

To examine how the model is progressing, the loss is plotted along the epochs, both for training and validation sets, as presented in Figure 4.2. It can be seen that in some cases, the loss increases from one epoch to another, which shows that a greater number of epochs doesn't mean that a better model accuracy will be obtained.

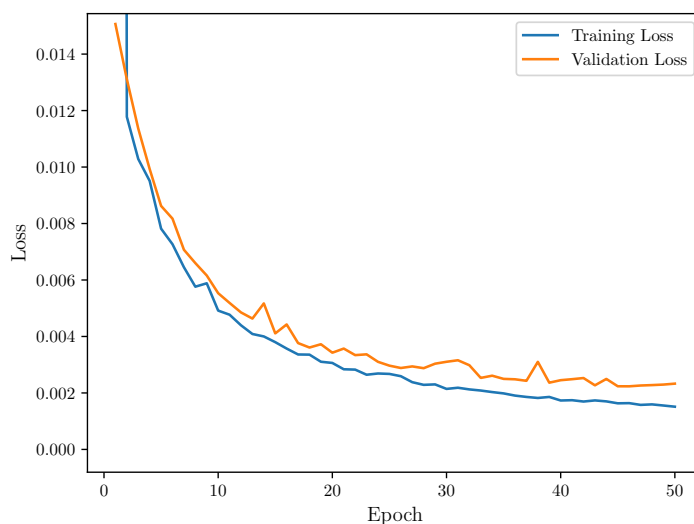


Figure 4.2: Example of the evolution of training and validation loss throughout the epochs.

After being successfully tested, the model must become available to being used in real applications, so for that it has to be deployed into a target platform, for which an interactive interface or API is usually created. To sum up this section, a list of steps that outline the process of developing a ML model will be introduced:

1. **Preprocess the data** that can be relevant for the defined problem.
2. **Split the data** into train, validation and test sets, choosing an appropriate split ratio for each set and applying data stratification.
3. **Choose a suitable model** and **train** it using the training set, after selecting an initial configuration of hyperparameters.
4. **Tune the hyperparameters** with the validation set, to adjust its architecture and prevent overfitting to the training set.
5. **Test** the final model on the training set, once again evaluating its generalization capabilities and performance on unseen data.
6. **Deploy** the ML model in a suitable environment or device.

4.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) have emerged as a result of inspiration drawn from biological neurons. Although ANNs are far from replicating exactly what these neurons do, they are both based on the same principle: individual neurons behave in a simple way, but when organized in a vast network, they are capable of performing highly complex computations.

The nodes (or neurons) in an ANN are compiled into subsequent layers and each layer can be of several types: dense layers, convolutional layers, recurrent layers, amongst others. Throughout this project, ANNs with dense layers will be adopted in all ML models, meaning that each node in layer l is connected to all nodes in layers $l - 1$ and $l + 1$ (Figure 4.3). The first and last layers of an ANN are called input and output layers, respectively, as the purpose of the first is to feed the input data into the model and the last presents the predicted output. Still regarding the architecture of an ANN, the number of nodes in a layer is associated to the width of the network, while the number of layers is related to its depth.

4.3.1 Perceptron

Based on the already mentioned work of McCulloch and Pitts, the concept of perceptron was developed in the 1950s, building the base for later development of artificial neurons. A perceptron only generates predictions that correspond to one of two states, usually $\hat{y} \in \{-1, 1\}$.

The perceptron is suitable to solve simple linear classification problems, by giving a binary output prediction \hat{y} after calculating the scalar product of the input data vector x with the weights vector w , and adding the scalar bias term b :

$$\hat{y} = \begin{cases} -1 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \quad (4.1)$$

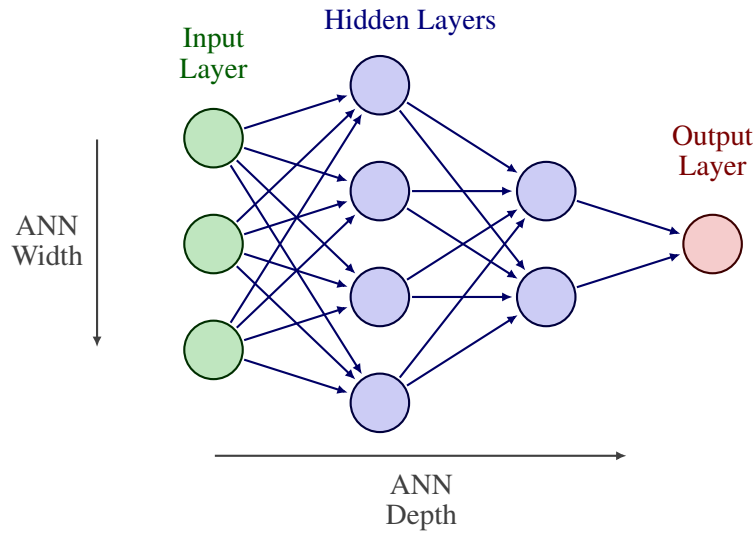


Figure 4.3: Example of an ANN composed only of dense layers, with two hidden layers. Adapted from (29).

Of course the predictions of a perceptron won't be accurate unless the weights and the bias are updated according to its behaviour, that is whether the prediction \hat{y}_i matches the label y_i or not. The Perceptron algorithm for updating these parameters is presented below in Algorithm 1. As inputs, it takes the dataset features X (matrix where each vector row x_i represents the features of one data instance) and labels y . In addition, the number of instances or dataset length n and the number of times T that the algorithm must iterate through the entire set must also be provided.

Algorithm 1 The Perceptron algorithm

```

procedure PERCEPTRON( $X, y, n, T$ )
   $w \leftarrow \{0\}$ 
   $b \leftarrow 0$ 
  for  $t = 1, \dots, T$  do
    for  $i = 1, \dots, n$  do
      if  $y_i(w \cdot x_i + b) \leq 0$  then
         $w \leftarrow w + y_i x_i$ 
         $b \leftarrow b + y_i$ 
      end if
    end for
  end for
end procedure

```

When the algorithm spots a mistake, i.e., $y_i(w \cdot x_i + b) = y_i \hat{y}_i \leq 0$, the updated values of the vector w and bias term b produce a better prediction if:

$$y_i[(w + y_i x_i) \cdot x_i + b + y_i] \geq y_i(w \cdot x_i + b). \quad (4.2)$$

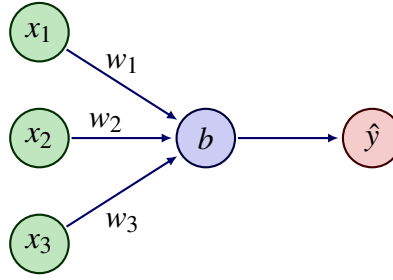


Figure 4.4: Scheme of a perceptron that takes 3 inputs.

The previous expression can be simplified into:

$$y_i^2 ||x_i||^2 + y_i^2 \geq 0, \quad (4.3)$$

which is always true, thus it is proved that the weights and bias updates always provide better predictions than their previous values.

4.3.2 Artificial Neuron

Compared to the original perceptron, an artificial neuron is slightly more complex and has the advantage of being able to predict not only categorical but also continuous values. It is based on the same principle that the output of a neuron is calculated by summing the weighted inputs from neurons in the previous layer. Then, an activation function f is applied to the resulting value and the final output is obtained. For the i -th node in the l -th layer, the node output is given by:

$$a_i^{(l)} = f \left(\sum_{j=1}^n w_{i,j}^{(l)} a_j^{(l-1)} + b_i^{(l)} \right), \quad (4.4)$$

where index j denotes for the node position in the previous $(l-1)$ layer. Figure 4.5 simplifies the understanding of this calculation method, by representing the first two layers of a generic ANN and focusing on the output of one specific node.

To summarize these equations in matrix notation, the computation of the outputs for all nodes in layer l can be presented as:

$$\begin{pmatrix} a_1^{(l)} \\ a_2^{(l)} \\ \vdots \\ a_m^{(l)} \end{pmatrix} = f \left[\begin{pmatrix} w_{1,1}^{(l)} & w_{1,2}^{(l)} & \cdots & w_{1,n}^{(l)} \\ w_{2,1}^{(l)} & w_{2,2}^{(l)} & \cdots & w_{2,n}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1}^{(l)} & w_{m,2}^{(l)} & \cdots & w_{m,n}^{(l)} \end{pmatrix} \begin{pmatrix} a_1^{(l-1)} \\ a_2^{(l-1)} \\ \vdots \\ a_n^{(l-1)} \end{pmatrix} + \begin{pmatrix} b_1^{(l)} \\ b_2^{(l)} \\ \vdots \\ b_m^{(l)} \end{pmatrix} \right] \quad (4.5)$$

$$a^{(l)} = f \left(W^{(l)} a^{(l-1)} + b^{(l)} \right) \quad (4.6)$$

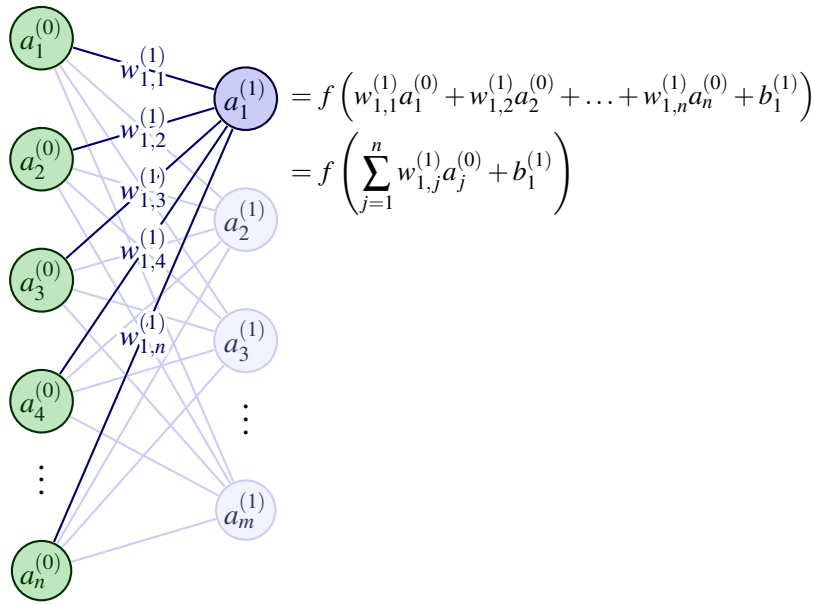


Figure 4.5: Activation of an artificial neuron. Adapted from (29).

Although the architecture is key for a good model behaviour, it must always be hand-in-hand with the choice of an appropriate activation function. These functions allow introducing non-linearity to the network, performing gradient propagation or limiting the node output. An ANN that has nonlinear activation, neurons connected to all the nodes in the previous layer and in the subsequent layer (fully-connected), and that is composed of input, hidden and output layers, is commonly referred to as a Multilayer Perceptron. However, sometimes there is no need to introduce such complexity in the network and a simple linear pass-through function can be used ($f(x) = x$), where the node output is simply the weighted sum of its inputs. Amongst the most common activation functions there are the Sigmoid, Rectified Linear Unit (ReLU) and Hyperbolic Tangent (tanh), which will be described below and plotted in Figure 4.6.

The Sigmoid is a differentiable function and it is a common choice when the model must predict a probability, as it outputs values between 0 and 1. It is calculated by:

$$f(x) = \frac{1}{1 + e^{-x}}. \quad (4.7)$$

ReLU turns negative values into zero and passes-through positive values. Nowadays, it is one of the most common activation functions and it can be expressed by:

$$f(x) = \max(0, x). \quad (4.8)$$

One of the disadvantages of ReLU is that all negative values are immediately mapped to 0, so its derivative is always null for negative values, which can have an unfavourable influence during model training, particularly at the backpropagation process. To solve this issue, the Leaky ReLU function was created. It multiplies values under 0 by a small constant (usually 0.01), so that the negative part of its graph has a small slope instead of being flat and their derivatives are not null

anymore. For a constant k such that $0 < k < 1$, the Leaky ReLU can be written as:

$$f(x) = \max(kx, x). \quad (4.9)$$

Similarly to the Sigmoid function, the tanh also limits the range of the outputs, but between -1 and 1, so this function has the advantage of mapping negative values into strongly negative outputs. It is given by:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (4.10)$$

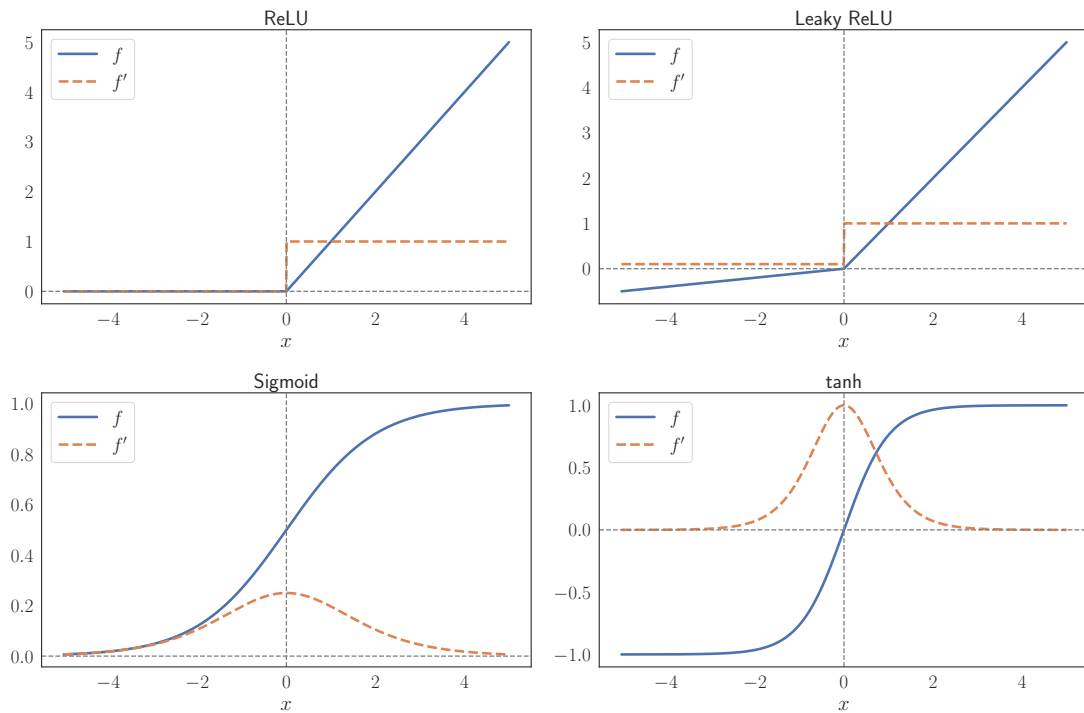


Figure 4.6: Activation function plots (blue) and their gradients (dashed orange). For illustration purposes, the Leaky ReLU constant is set to 0.1, instead of the usual 0.01.

The gradients of activation functions play a significant role in the model training, as each step to update the weights and bias requires their calculation. The update of these parameters is proportional to the partial derivatives being computed, so if their value is too small, the updates will be minimal and may even be null in limit cases. This issue is known as vanishing gradient, and applying activation functions as the Sigmoid or tanh can lead to this problem, as their gradient is almost null for a large portion of their domain. Hence, throughout the development of ML models in this project, ReLU and Leaky ReLU were the adopted functions.

So far, we have seen that the network receives the inputs in the first layer and how they flow sequentially through the layers of the network (applying the activation function to the weighted sum of the inputs in each layer), until an output prediction is generated. This process of input data

propagation until the final prediction is called a forward pass. Next, the metrics used to study the accuracy of the model after the forward pass will be described.

4.3.3 Loss Functions

An ANN is reliable if its predictions are equal or very close to the real outputs or labels associated with a specific instance. Predictions are evaluated through loss functions and the algorithm that optimizes the network weights and bias, does so based on setting the minimization of the loss function as an objective.

There are many loss functions that can be used according to each problem, so only the most used in regression problems with continuous values will be presented.

The Mean Squared Error (MSE or L2 Loss) is the most common loss function for regression problems. This function averages the squared difference between predicted and actual outputs, so large errors are strongly penalized by being squared, making it less consistent to deal with outliers. It is expressed by:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (4.11)$$

where n is the length of the dataset, \hat{y}_i represents the predicted output for instance i and y_i its actual value.

One alternative that is more robust to outliers is the Mean Absolute Error (MAE or L1 Loss). Nevertheless, it does not emphasize the reduction of large errors when compared to small ones. It is formulated as:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (4.12)$$

To merge the advantages of the previous functions, the Huber Loss was created. A threshold t is defined, so if the absolute difference between the prediction and the actual value is greater than t , the MAE is applied, if not, the MSE is used:

$$L_t = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \leq t \\ t|y - \hat{y}| - \frac{t^2}{2} & \text{otherwise} \end{cases}. \quad (4.13)$$

Figure 4.7 compares the loss obtained applying the three described methods, for the same error $(\hat{y}_i - y_i)$, taking the threshold value of the Huber Loss as $t = 2$.

4.3.4 Backpropagation

Backpropagation is one of the fundamental algorithms used in ANNs, allowing the network to update its weights and bias to increase the model performance. The backpropagation algorithm can be divided in two steps: forward step and backward step.

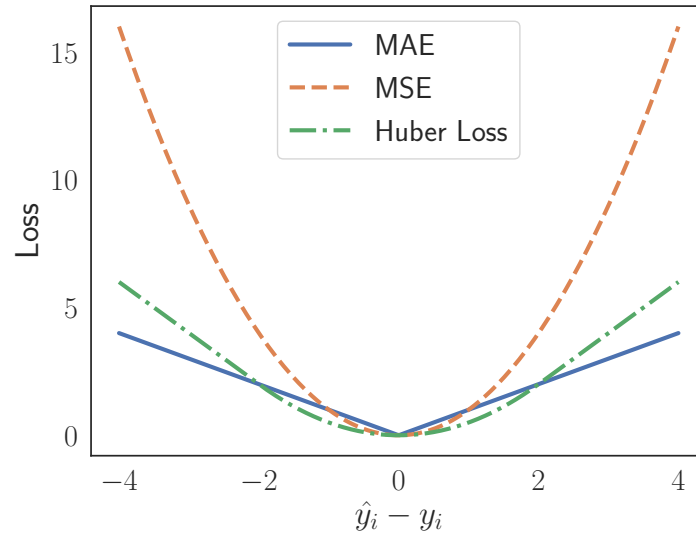


Figure 4.7: Comparison between loss functions, considering $t = 2$ for the Huber Loss.

The forward step essentially corresponds to everything that was previously described in this section. First, the weights and bias are initialized, the network takes the inputs and the forward pass is executed, concluding with the model predictions.

From that point, the backward step starts. The gradient of the loss function with respect to the model output is calculated and then backpropagated through the network, computing the loss gradient in relation to the weights and bias from the output to the input layer. At last, an optimization algorithm is employed to update the network parameters so that the loss value decreases in each iteration.

The equations used in the backpropagation algorithm will be presented below and the simple scheme in Figure 4.8 intends to ease the understanding of how these equations were derived and the notation used. Only one node is represented in each layer and a greater focus is given to the nodes in layer l and in the output layer L .

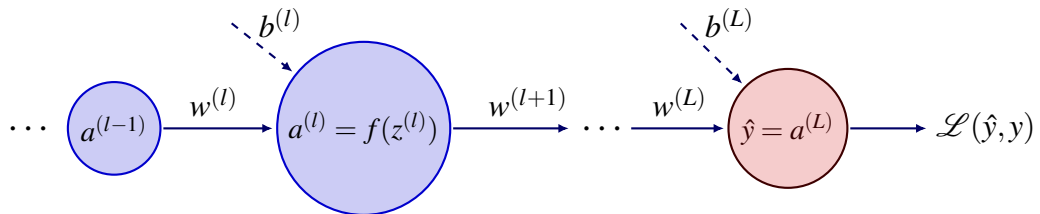


Figure 4.8: Simple ANN with only one node in each layer. Only the nodes in layers $l - 1$, l and in the output layer L are represented.

The vector representing the values of all neurons in layer l of a given ANN, $a^{(l)}$, has been already characterized in this section, but it is necessary to define a new variable, corresponding to the weighted inputs of the nodes in layer l , $z^{(l)}$, that is expressed by:

$$z^{(l)} = w^{(l)} \cdot a^{(l-1)} + b^{(l)} \quad (4.14)$$

Considering a generic activation function f (whose derivative f' is known), it is clear that $a^{(l)} = f(z^{(l)})$. The objective of backpropagation is to compute the partial derivatives of the loss function \mathcal{L} in order to the weights w and the bias b : $\partial \mathcal{L} / \partial w$ and $\partial \mathcal{L} / \partial b$, respectively. Another variable must be introduced, translating the error of the i -th neuron in layer l , $\Phi_i^{(l)}$ (or simply $\Phi^{(l)}$ in vector notation):

$$\Phi_i^{(l)} = \frac{\partial \mathcal{L}}{\partial z_i^{(l)}}, \quad (4.15)$$

applying the chain rule, it is equivalent to saying that:

$$\Phi_i^{(l)} = \frac{\partial \mathcal{L}}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}}, \quad (4.16)$$

besides, as the relation between a and z is established, the previous equation can be rewritten as:

$$\Phi_i^{(l)} = \frac{\partial \mathcal{L}}{\partial a_i^{(l)}} f'(z_i^{(l)}). \quad (4.17)$$

To simplify computation during backpropagation, it may be expressed in matrix-based form, by:

$$\Phi^{(l)} = \nabla_a \mathcal{L} \odot f'(z^{(l)}), \quad (4.18)$$

where $\nabla_a \mathcal{L}$ is a vector with the partial derivatives $\partial \mathcal{L} / \partial a_i^{(l)}$ and \odot is the Hadamard product or element-wise multiplication. Now the expression to calculate the error Φ in all nodes of a given layer l has been deduced, but it has not been presented as a function of the error in the next layer, which is crucial to perform the backward step.

Going back to Equation 4.15, the chain rule can be once again applied, but now to introduce a variable from the next layer $l + 1$:

$$\Phi_i^{(l)} = \sum_k \frac{\partial \mathcal{L}}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = \sum_k \Phi^{(l+1)} \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}}, \quad (4.19)$$

with k denoting the node position in layer $l + 1$. The term $\partial z_k^{(l+1)} / \partial z_i^{(l)}$ is evaluated with the help of Equation 4.14 depicted in component form:

$$z_k^{(l+1)} = \sum_i w_{ki}^{(l+1)} a_i^{(l)} + b_k^{(l+1)} = \sum_i w_{ki}^{(l+1)} f(z_i^{(l)}) + b_k^{(l+1)}. \quad (4.20)$$

Hence, that term is given by:

$$\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = w_{ki}^{(l+1)} f'(z_i^{(l)}) . \quad (4.21)$$

Replacing it in Equation 4.19, we arrive at:

$$\Phi_i^{(l)} = \sum_k \Phi_k^{(l+1)} w_{ki}^{(l+1)} f'(z_i^{(l)}) , \quad (4.22)$$

or in matrix form:

$$\Phi^{(l)} = ((w^{(l+1)})\Phi^{(l+1)}) \odot f'(z^{(l)}) . \quad (4.23)$$

Finally, to reach the goal of computing $\partial \mathcal{L} / \partial w$ and $\partial \mathcal{L} / \partial b$, the same reasoning can be applied, but this time what is obtained by introducing weights or bias related terms to Equation 4.15 through the chain rule is much simpler, and the relation between z and w or b given in Equation 4.14, leads to:

$$\frac{\partial \mathcal{L}}{\partial b_i^{(l)}} = \Phi_i^{(l)} \quad (4.24)$$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} = a_j^{(l-1)} \Phi_i^{(l)} \quad (4.25)$$

To conclude, the entire backpropagation process can be summarized in the following steps:

1. **Initialize** weights and bias randomly and define the inputs.
2. Sequentially compute $z^{(l)}$ and $a^{(l)}$ for each layer l of the network, until the final layer is reached and a prediction is generated (**feedforward**).
3. For the output layer L , **calculate the error** $\Phi^{(L)}$ through Equation 4.18.
4. **Backpropagate the error**, from the last layer to the initial, using Equation 4.23.
5. **Evaluate the gradient of the loss function** in order to the bias and weights, with Equations 4.24 and 4.25.

4.3.5 Optimization Algorithms

To complement backpropagation, the bias and weights of the network must be updated according to the loss function gradients, so that the ANN performance improves and the final loss value is minimized. These updates are carried out using optimization algorithms, for which an objective function must be defined as well as the information of whether this function should be maximised or minimised. Typically the objective of the optimization is set as:

$$\min \left(\mathcal{L}(\theta) = \frac{1}{T} \sum_{t=1}^T \mathcal{L}_t(\theta) \right) , \quad (4.26)$$

where T is the number of times the algorithm runs through the dataset and θ represents the parameters to be optimized (usually the weights and bias for an ANN).

There are currently dozens of optimization algorithms that are regularly applied to ML frameworks, however only the main algorithms for ANN regression problems will be covered, as they are the most suited for this project. Going forward, the Stochastic Gradient Descent and Adaptive Moment Estimation algorithms will be briefly introduced.

The Stochastic Gradient Descent (SGD) is very often used as an optimization algorithm for ML purposes. It was first presented in 1951 (43), but has been improved over the years. The foundations of SGD rely on the Gradient Descent (or Batch Gradient Descent, BGD) algorithm, with some simplifications that steeply increase its computational efficiency. As the BGD requires going through the entire training set to update the parameters, SGD only uses portions of this set for each update, called mini-batches, making it much faster than BGD.

A simplified version of the SGD is presented in Algorithm 2, where η is the learning rate, a variable defining the step size at which the parameters θ are updated. However, other parameters besides the presented ones have been introduced to this method, such as weight decay, that prevents overfitting, and momentum, that accelerates convergence (46).

Algorithm 2 Simplified SGD algorithm

```

procedure SGD( $\theta_0, \eta, T, \mathcal{L}$ )
  for  $t = 1, \dots, T$  do
     $g_t \leftarrow \nabla_{\theta} \mathcal{L}_t(\theta_{t-1})$ 
     $\theta_t \leftarrow \theta_{t-1} - \eta g_t$ 
  end for
end procedure

```

Another much more recent optimization algorithm is the Adaptive Moment Estimation, commonly known as the Adam algorithm. As opposed to SGD that keeps the learning rate unchanged throughout its iterations, the Adam algorithm computes learning rates for each parameter and adapts them based on the previous gradients of those parameters (18). This is done with the use of 2 variables referred to as the first and second moments of the gradients, respectively u and v . The first moment tracks the mean direction of the gradients, while the second keeps record of their variance. Adam was presented as an improvement of algorithms that would also be appropriate for regression frameworks, such as AdaGrad and RMSProp, specially in terms of generalizing, which translates the model's accuracy to predict the output of instances that it hasn't trained on.

Algorithm 3 presents a simple adaptation of Adam, which relies only on the strictly needed inputs. A weight decay parameter could also be added and other versions of this method could be applied, such as the AMSGrad algorithm (41), that modifies the update rule for the second moment.

Algorithm 3 Simplified Adam algorithm

```

procedure ADAM( $\theta_0, \eta, T, \mathcal{L}, \beta_1, \beta_2$ )
   $u_0 \leftarrow 0$ 
   $v_0 \leftarrow 0$ 
  for  $t = 1, \dots, T$  do
     $g_t \leftarrow \nabla_{\theta} \mathcal{L}_t(\theta_{t-1})$ 
     $u_t = \beta_1 u_{t-1} + (1 - \beta_1) g_t$ 
     $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
     $\hat{u}_t = u_t / (1 - \beta_1^t)$ 
     $\hat{v}_t = v_t / (1 - \beta_2^t)$ 
     $\theta_t \leftarrow \theta_{t-1} - \eta \hat{u}_t / \sqrt{\hat{v}_t + \epsilon}$ 
  end for
end procedure

```

Comparing these two methods, it become evident that the SGD has the advantage of being simpler, while still performing well. On the other hand, it presents slower convergence in some cases and it is very sensitive to the choice of the learning rate, as it always remains constant. Overall, the performance of Adam is usually better, since the adaptive learning rates accelerate the convergence, however it is worst in generalizing the model, being more prone to overfitting. Thus, both algorithms will be tested when developing the successive iterations of ANN models in this work.

4.4 Implementation in PyTorch

Throughout this section, simplified PyTorch implementations of some of the processes that were previously described will be presented. PyTorch allows to develop ML frameworks in a simple way, since its many essential built-in functions perform the longest and more complex calculations, so, for instance, there is no need to explicitly program all backpropagation equations nor to write down all the steps of the optimization algorithms. To reduce the depicted code, it will be assumed that the libraries and modules imported for each topic will be already imported in the following topics. Another assumption made is that the hyperparameters (represented by uppercase variables) have been previously defined, so their values are not shown.

Train-Validation-Test split with stratification

There are some functions in the ML library *scikit-learn* that allow performing the split and stratifying the data at the same time, however they all rely on categorical outputs and the aim of this framework is to predict continuous variables. Thus, the target variables y must be combined into a certain number of bins (exemplified as 10), which are representative of a given output range. The first split sets aside the test dataset, while the second distinguishes training from validation sets. As the second split is only using a part of the original dataset, the validation ratio must be adjusted so that its defined value (15%) is given in relation to the entire dataset.

```
import pandas as pd
from sklearn import train_test_split

test_ratio = 0.15
valid_ratio = 0.15
valid_ratio = valid_ratio/(1-test_ratio)
bin_count = 10

bins = pd.qcut(y, bin_count, labels=False, duplicates='drop')
x_train, x_test, y_train, y_test = train_test_split(
    x, y,
    test_size=test_ratio,
    stratify = bins)
bins = pd.qcut(y_train, bin_count, labels=False, duplicates='drop')
x_train, x_valid, y_train, y_valid = train_test_split(
    x_train, y_train,
    test_size=valid_ratio,
    stratify = bins)
```

Dataset class and DataLoader

To load the dataset into a PyTorch ML model, the class *DataLoader* is of great relevance. It allows defining the batch size or if whether or not the data must be shuffled, amongst other parameters. To complement the *DataLoader*, a user-defined class is usually built, to differentiate the features from the targets, as it is the case of the *Hertz_Dataset* class.

```
from torch.utils.data import DataLoader

class Hertz_Dataset():
    def __init__(self, features, targets):
        self.features = features
        self.targets = targets
    def __len__(self):
        return len(self.targets)
    def __getitem__(self, idx):
        return self.features[idx], self.targets[idx]

train_data = Hertz_Dataset(x_train, y_train)
test_data = Hertz_Dataset(x_test, y_test)
```



```

valid_data = Hertz_Dataset(x_valid, y_valid)

train_loader=DataLoader(train_data,
                        batch_size=BATCH_SIZE,
                        shuffle=True)
test_loader=DataLoader(test_data, shuffle=False)
valid_loader=DataLoader(valid_data, shuffle=False)

```

Create and instantiate the model

First, the ANN model must be created based on the subclass *nn.Module*, that contains all the building blocks that are needed for neural networks. The example given below is for the regression problem of trying to predict Young's modulus of a sample, based on its F-I approach curve, so there is only one output. As an input, there is a matrix with two columns, for indentation and force in each point of the curve, that must be flattened to acquire the correct shape to be introduced to the model's first layer, as schematized in Figure 4.9. The size of the input layer is computed based on the number of elements in the original matrix.

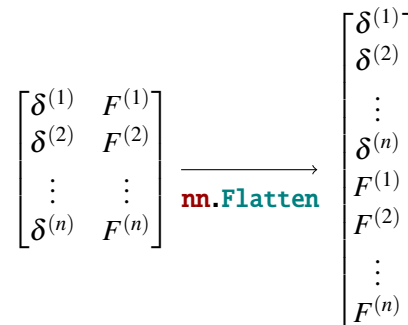


Figure 4.9: Flattening the inputs.

All the layers are combined inside *nn.Sequential* and the example presented has only one hidden layer, using the ReLU activation function. Every subclass of *nn.Module* needs to override *forward()*, so that the forward step is performed. Finally, the model is attributed to an instance and the loss function and optimization algorithm are defined.

```

import torch
from torch import nn

# Define the ANN model
class Regression_Hertz(nn.Module):
    def __init__(self, input_shape, HIDDEN_UNITS):
        super(Regression_Hertz, self).__init__()

```

```

input_size = input_shape[0] * input_shape[1]
self.layers = nn.Sequential(nn.Flatten(),
                             nn.Linear(input_size, HIDDEN_UNITS),
                             nn.ReLU(),
                             nn.Linear(HIDDEN_UNITS, 1))

def forward(self, x):
    out = self.layers(x)
    return out

# Instantiate the model
input_shape = x_train.shape[1:]
model_Hertz = Regression_Hertz(input_shape, HIDDEN_UNITS)

# Define the loss function and optimizer
loss_fn = nn.MSELoss()
optimizer = torch.optim.Adam(model_Hertz.parameters(),
                              lr=LEARNING_RATE)

```

Train one epoch

A function to train one epoch was created, to later be embedded in the training and validation loop for all the epochs. The data in the training set is fed to the model in a "for" cycle, where a batch is loaded in each iteration. After getting the batch data, the first step to take is to zero the gradients, as PyTorch accumulates its previous values by default. Then, the forward and backward steps are computed, and the optimizer updates the parameters. This simplified function returns the loss value for each prediction, that will be useful to later analyse the model's behaviour.

```

def train_one_epoch(train_loader):
    loss_list = []
    for _, data in enumerate(train_loader):
        # Every data instance is an input + target pair
        inputs, targets = data
        # Zero the optimizer gradients
        optimizer.zero_grad()
        # Performs the forward step
        outputs = model_Hertz(inputs)
        # Compute the loss and its gradients
        loss = loss_fn(outputs, targets)
        # Backpropagate the gradients

```

```

    loss.backward()
    # Adjust parameters
    optimizer.step()
    # Gather data and report
    loss_list.append(loss.item())
return loss_list

```

Train-Validation loop

Below there is a condensed version of the train-validation cycle in which the previous function for training one epoch is incorporated. At the beginning of each epoch, that function is called, allowing the model to be trained, and then the loss associated with each training instance is recorded.

It is then studied how accurate are the model's predictions on the validation set, and in the end the model is saved with the parameters that allowed for the minimum value of validation loss. In the full code of this framework, a more complete version of the code presented is built inside a function that not only saves all the relevant training and validation information for each epoch, but also produces essential plots that monitor the model's performance.

```

best_vloss = 1e6
train_loss = []
for epoch in range(EPOCHS):
    # Train one epoch
    loss_list = train_one_epoch(epoch, train_loader)
    train_loss.append(loss_list)
    # Evaluation on the validation set
    running_vloss = 0.0
    for i, vdata in enumerate(valid_loader):
        vinputs, vtargets = vdata
        vpredicts = model_Hertz(vinputs)
        vloss = loss_fn(vpredicts, vtargets)
        running_vloss += vloss
    avg_vloss = running_vloss/(i + 1)
    # Track best performance
    if avg_vloss < best_vloss:
        best_vloss = avg_vloss
        model_path = 'model_state_dict_{}.pt'.format(epoch+1)
torch.save(model_Hertz.state_dict(), model_path)

```

Load and test the model

To test the saved model, it is first necessary to load it. An instance is created with the architecture of the model and then the parameters that were saved in the end of training and validating are assigned to it. The model is turned into test mode and its performance is analysed over the test set. Once again, in the code developed for this project, a more complex version of this "for" loop is used inside a function that returns a significant amount of relevant information about the model, either as numeric values or as graphs.

```
loaded_model = Regression_Hertz(input_shape, HIDDEN_UNITS)
# Load in the saved parameters
loaded_model.load_state_dict(torch.load(f=model_path))

loss = 0
inputs_list, targets_list, predicts_list = [], [], []

# Test the model
loaded_model.eval()
with torch.inference_mode():
    for i, testdata in enumerate(test_loader):
        test_inputs, test_targets = testdata
        inputs_list.append(test_inputs)
        targets_list.append(test_targets)
        y_predicts = loaded_model(test_inputs)
        predicts_list.append(y_predicts)
        loss += loss_fn(y_pred, test_targets)
    avg_loss /= len(test_loader)
```

Chapter 5

Data Generation and Model Development

Throughout this chapter, a detailed description will be given of all the steps taken to develop the Machine Learning models from scratch, starting with the creation of synthetic data to later arrive at the choice of the final models. Two ANN models were built, to predict surface properties (Young's modulus E and adhesion energy γ) from AFM synthetic F-I curves. The parameters of both models were carefully optimized to enhance their performance, for later being fit to predict these properties based on experimental data.

5.1 Synthetic Data Generation

Before starting the development of a Machine Learning model, a suitable dataset must be defined. But there are common problems when choosing the right data or even trying to get access to it. For some cases, the available data might be deeply insufficient, while for others there might be privacy concerns at stake, so it is important that the model doesn't fully replicate the original data. To overcome these limitations, it has become common practice to train a ML model that from synthetically generated data can represent the underlying patterns of a real dataset. In addition, it gives more control over the characteristics of the data, thus allowing to study narrow and specific scenarios, if needed.

Many fields have been focusing on how to improve their synthetic data to produce better models. It has been thoroughly studied for face recognition (6), as it faces ethical concerns on the use of authentic biometric data. Medicine and health sciences in general can also benefit from this approach, that prevents from having to use patient-specific information and eases the process of data sharing (40). This method can be also of key importance in AFM related ML models, as data related to this type of analysis is usually scarce. For instance, synthetic data reproducing the interaction between tip and sample has been created to serve as the basis for identifying vibration mode coupling in dynamic AFM (2). In topographic modes, this method is also useful to generate

AFM images with different kinds of distortions for then producing a model that identifies and corrects typical AFM artifacts (19).

For this framework, AFM force-indentation curves were generated, mimicking the behaviour of soft biological samples, both for approach and retraction stages, using the contact models described in chapter 3. Hertzian theory-based Equation 3.15 coupled with Equation 3.17 were the foundation to compute the force for the approach stage, while Equation 3.28 was used for the withdraw phase. It is thus important to define what are the fundamental variables that will affect the forces for each stage (F_{Hertz} and F_{JKR}), so it can be written that:

$$F_{Hertz} = F_{Hertz}(\delta, E, \nu, R) \quad (5.1)$$

and

$$F_{JKR} = F_{JKR}(\delta, E, \gamma, \nu, R) . \quad (5.2)$$

So all these variables must be defined to reach the final curves. The Poisson ratio and the tip radius will be the same for all curves, while the other parameters are unique for a specific curve. Thus, each indentation vector will have material parameters assigned to it, and jointly with R and ν will allow to compute the force for all indentation values in the vector. Knowing the force and indentation for each point, the curves can be obtained.

Starting with the Poisson ratio, as the target of this model are soft tissues, they are usually modeled as incompressible, due to their high-water content, and incompressibility corresponds to a Poisson ratio of 0.5. The tip radius value, presented in Table 5.1, was chosen according to the probe that was used to produce the experimental nanoindentations that will be later used to test the developed models, and it is within a common range of values for spherical tips radii.

Several vectors were created, characterizing the indentation depth from the start to the end of a nanoindentation cycle. Negative indentation values (when tip-sample contact is not established) were also considered, but solely for illustration purposes, since only its positive values (describing tip-sample contact) are relevant for the analysis. Regarding indentation depth vectors, it is important to define the maximum indentation and the number of positive values, which will define the number of points for each curve and consequently set the input shape of our models. For simplification purposes, $\delta_{contact}$ will be defined as the vector of positive indentation values for an half-cycle. For the same F-I curve, approach and retraction indentation vectors present the same values, but in a mirrored way, as depicted below:

$$\delta_{approach} = [\delta^{(0)}, \delta^{(1)}, \dots, \delta^{(n)}], \delta_{withdraw} = [\delta^{(n)}, \dots, \delta^{(1)}, \delta^{(0)}] . \quad (5.3)$$

In order for the indentation vectors to resemble real data more closely and to have a greater impact on the ML models, they can't all be composed of the same indentation values. If this was the case, our models wouldn't learn anything based on indentation, since different curves would always have the same indentation values and would only differ in the tip-sample forces. Hence,

Table 5.1: Initial variables definition for synthetic data generation.

Parameter	Value
ν	0.5
R	1980 nm
$\delta_{contact}$	Max. indentation: 150 nm Length: 50

apart from a first contact vector with evenly spaced values from 0 to the maximum indentation, all the remaining were created taking advantage of NumPy's *random* library, as depicted in the following code, where the variable *size* represents the number of instances in the entire synthetic dataset.

```

import numpy as np
# Define maximum indentation and vector size
xmax, npts = 150, 50
# Create a first vector with evenly spaced values
contact = np.linspace(0, xmax, npts+1)
rnd_contact_list = [contact]
# Append to the list the remaining vectors with randomly spaced values
for _ in range(size-1):
    rnd_contact = np.random.random(npts+1).cumsum()
    rnd_contact = (rnd_contact-rnd_contact.min()) / rnd_contact.ptp()
    rnd_contact = (xmax-0)*rnd_contact
    rnd_contact_list.append(rnd_contact)

```

After specifying the Poisson ratio, tip radius, and a set of displacement vectors, the only remaining parameters are the material properties E and γ , which must also be comprised in a typical range of values for soft tissues.

For the Young's modulus, its values commonly fluctuate between the orders of magnitude of 10^{-1} to 10^1 kPa, reaching the hundreds of kPa in more specific scenarios (31)(24). Taking also into account the Young's moduli of the samples in the experimental data available, a maximum of 10 kPa was fixed.

On the other hand, our experimental dataset does not contain any information about surface energy, since the available retraction curves are scarce and the corresponding samples didn't necessarily exhibit strong adhesion properties. Hence, its range of values was searched in existing literature regarding biological samples (56)(50)(26). The values found go from hundreds of $\mu\text{J}/\text{m}^2$ to the dozens of mJ/m^2 , making it difficult to establish a shorter interval. In addition, the samples in the available experimental dataset didn't necessarily all have adhesive properties with high values, as the ones in the works mentioned. Thus, we sought to get a range of values that would

somehow illustrate the relationship between the approach and retraction curves, considering the Young's moduli range found in the dataset. For instance, if a set of γ values around 0.1 mJ/m^2 was defined, an unrealistic representation of AFM nanoindentations would be obtained, as depicted in Figure 5.1, where we can see that the withdraw curves are not near the approach ones, in the contact region. For greater adhesion energies, the discrepancy would be even more pronounced.

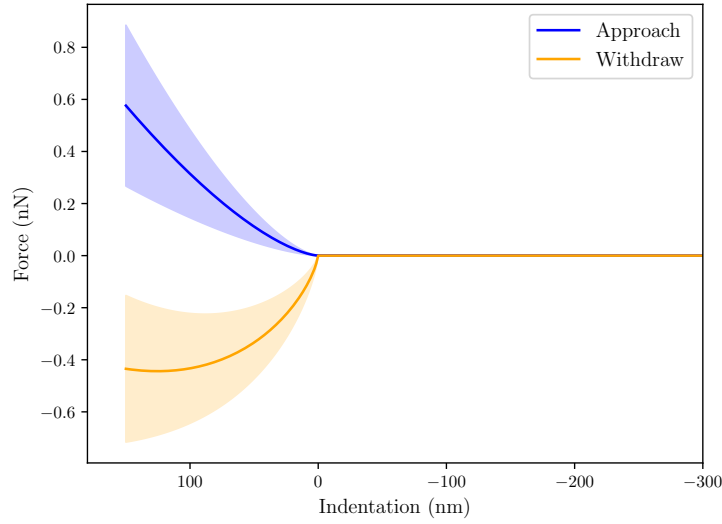


Figure 5.1: Unrealistic representation of AFM nanoindentation curves ($E \in [0.1-10] \text{ kPa}$ and $\gamma \in [10-100] \text{ } \mu\text{J/m}^2$). The lines represent the mean and the bands the standard deviation.

Hence, this parameter was studied in the lower range of $1 - 3 \text{ } \mu\text{J/m}^2$. Figure 5.2 illustrates the relation between approach and retraction curves for the final material parameters range, that is summarised in Table 5.2.

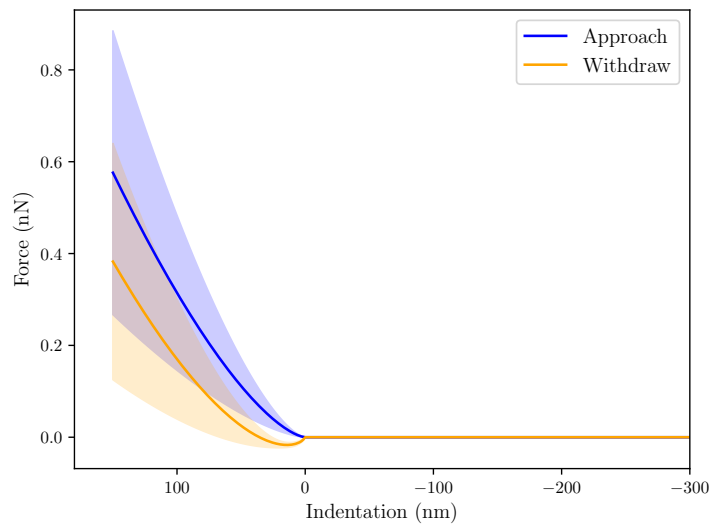


Figure 5.2: Distribution of the synthetic approach and withdraw curves for the initial dataset.

Table 5.2: Ranges of values for material properties.

Material property	Min.	Max.
E	0.2 kPa	10 kPa
γ	1 $\mu\text{J}/\text{m}^2$	3 $\mu\text{J}/\text{m}^2$

When generating values for these material parameters, it is also important to carefully consider the choice of the distribution function, since it will significantly impact the realism and representativeness of the synthetic data generated. In addition, it is beneficial to the model that it is trained on data with a distribution similar to a corresponding real or experimental dataset. This way it can focus more deeply in the most frequent values, and doesn't "waste" a great amount of time training on instances with less frequent targets, that will have only a small impact in the testing performance.

For the stiffness parameter in biological cells, hydrogels or soft polymers in general, it is usually closer to the lower end of our range (roughly between 0.5 kPa and 5 kPa), so a triangular distribution, with the mode set to be inside the most frequent interval of values, would be appropriate to start training our model.

Regarding surface energy, as its values frequency is currently not as well established as in the case of Young's modulus, an uniform distribution will be the first approach for the retraction curves model, to get a better assessment of the influence of the extreme values in our range.

At last, the dataset size must be carefully chosen. This will play a critical role in the development of our models, as insufficient data can lead to poor model performance and overfitting, since there is not enough information to recognize patterns in the data, so the model only "memorizes" it. In contrast, if the dataset is too extensive, its computational complexity increases, leading to much higher training times. In addition, there is also the chance of the model overfitting to noise in the data. In common ML applications the number of instances in a dataset usually goes from the few thousands to the hundreds of thousands, and it is no different when applied to AFM frameworks (30)(52). Thus, to ensure a robust initial model without excessively compromising computational costs, an initial dataset consisting of 40 000 curves was created.

For a more accurate study into the impact of a specific hyperparameter on the model's behaviour, it is essential to train models with different settings on the same dataset. This ensures that any observed variations in performance are solely attributed to the hyperparameter under investigation, rather than being influenced by differences in the training data. Despite the material parameters being created with NumPy's *random* library, there is no need to save the initial dataset and constantly reload it to study models with different parameters. All it takes is to "plant" a *random.seed* (by assigning to it an integer number), as depicted in the code below, that makes the distribution functions produce always the same values for a particular seed. Figure 5.3 shows the initial distribution of the material parameters, generated by the shown code, with the random seed specified.

```

size = 40 000
np.random.seed(42)

# Triangular distribution for E values
E = np.random.triangular(left=0.2, mode=1.8, right=10, size=size)
# Uniform distribution for gamma values
gamma = np.random.uniform(low=1, high=3, size=size)

```

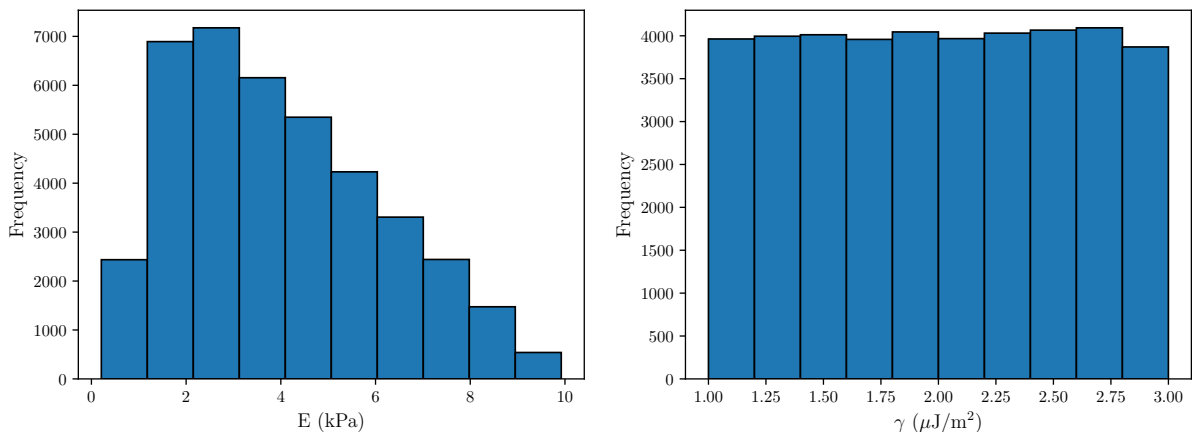


Figure 5.3: Initial distribution of the material parameters E and γ .

To allow reproducibility of the results presented in the following sections, a random seed was set for all the functions and methods that require any sort of randomness. Not only it was applied to the generation of material parameters, but also to the indentation vectors creation. It is also present in tasks like the train-validation-test split, ensuring the data in each set is always the same, and in the initialization of the ML model's parameters, so that the initial weights and bias don't influence the analysis. In all these cases, the random seed parameter was defined as 42. With an exception, of course, to when the models' performance is being tested over different splits or testing sets.

Finally, a set of synthetic nanoindentation curves is shown in Figure 5.4, illustrating their behaviour for extreme E and γ configurations. The main difference from these curves to the experimental ones is that the detachment point is coincident with the point where the contact is established in the approach cycle. A proper correction would be required, by determining a detachment point for each curve, which is still one of the main challenges in AFM. This won't have any serious negative impact on testing the model for the approach curves with experimental data. As for retraction curves, since they're being analysed separately from the approach cycle, the model that predicts their material parameters will be able to capture their shapes, thus enabling its generalisation for future applications where the zero indentation point computation could be incorporated.

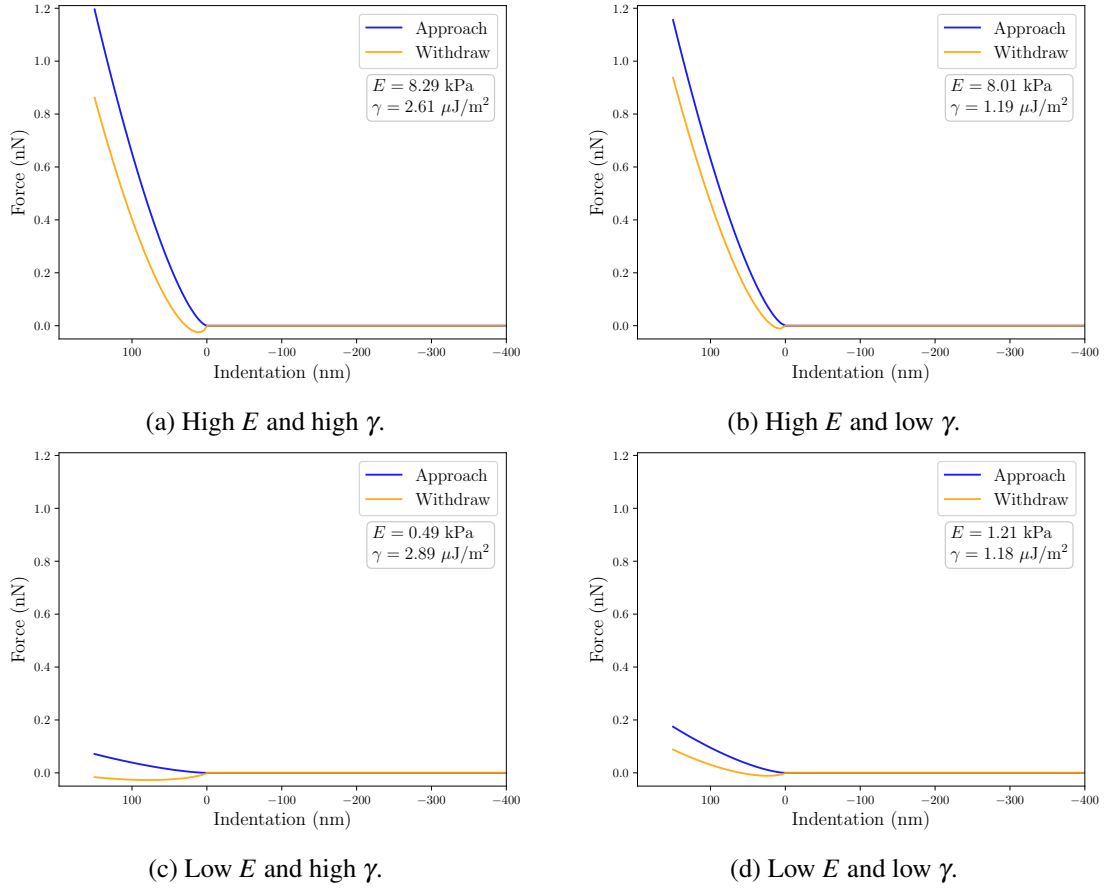


Figure 5.4: Synthetic nanoindentation curves in the initial dataset, for distinct E and γ configurations.

In the following sections, the ANN models development will be explained step by step, from their initial configuration to the procedures that led to the choice of the final hyperparameters. Each ML model will be identified based on the contact theory used to generate the F-I curves and both will have Multilayer Perceptron (MLP) architectures, where both the inputs and the outputs will be continuous values and they will be composed of dense linear layers. This type of architecture is common in problems as the one presented in this work, where there is lack of knowledge to guide or even justify the development of craftier architectures. This stands in contrast to more frequent Deep Learning problems, like the ones related to computer vision, where Convolutional Neural Networks are well established, or natural language processing, where Recurrent Neural Networks are often the preferred choice.

Concerning the developed models, at first there is the Hertz MLP, that will take the approach curves as inputs, to predict one single parameter: the Young's modulus. The second will be the JKR MLP, that will predict the Young's modulus and the adhesion energy, using the retraction curves.

5.2 Development of the Hertz MLP

Our first model will consist of a MLP and it can also be described as a Fully Connected Neural Network (FCNN), since the outputs of all neurons in one layer are connected to all neurons in the subsequent layer.

Before moving on to studying different steps in the model development and tuning the hyperparameters, an initial configuration will be defined, being improved with the successive iterations. This base ANN will be composed of 2 hidden layers, the first with 256 nodes and the second with 64, to provide some strength to the model. It will be trained for 20 epochs, with a learning rate of 10^{-3} and a batch size of 32, which are common values for initial assessments of ML models.

ReLU was chosen as the activation function and the loss was computed with the Mean Squared Error, while Adaptive Moment Estimation (Adam) was set as the first optimizer. The model will be trained with 70% of the approach curves in the entire dataset, while the other 30% are equally split for validation and testing.

Table 5.3: Hyperparameters of the baseline Hertz MLP model.

Hidden Layers (HL)	Nodes HL 1	Nodes HL 2	Epochs	Learning Rate	Batch Size
2	256	64	20	10^{-3}	32

Table 5.4: Initial activation and loss functions, optimization algorithm and split ratios for the Hertz MLP.

Activation	Loss	Optimizer	Data split		
			Train	Validation	Test
ReLU	MSE	Adam	70%	15%	15%

To get a better understanding of the scale of the loss values and their real meaning in the comparison of predicted and actual Young's moduli, the Mean Absolute Percentage Error (MAPE) will be used along this work, being often shortened to simply "Error (%)" or represented by $\bar{\varepsilon}$. This will serve as the foundation for another metric that will be consistently employed throughout the work, which is to state the percentage of instances, in this case F-I approach curves, whose Young's moduli were predicted with an error ε (relative error for an individual instance) below certain threshold values, generally 2.5%, 5% and 10%. The MAPE or $\bar{\varepsilon}$ is calculated by:

$$\bar{\varepsilon} = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (5.4)$$

Throughout this section, we'll first study the influence of stratification and define a suitable dataset split ratio. After that, the most appropriate loss and activation functions will be examined. At the end, Optuna (a framework for ML hypertuning) will be implemented, allowing to fine tune the remaining hyperparameters of the MLP.

5.2.1 Stratification and Split Ratio

Using the previously presented hyperparameters and model functions for the base ANN, the importance of stratifying the data was analysed, by comparing the model's performance with and without a stratified split. For the Hertz MLP, stratification was done as explained in section 4.4, splitting the target variable into 100 bins. In Figure 5.5a, there is a representation of the distribution of the target variable over the different sets, when stratification is applied, while Figure 5.5b depicts a random split without stratification. Even with a random split, it is unlikely that the data distribution will differ significantly between the subsets. This is because the initial dataset follows a well-established triangular distribution, and the resulting subsets from the split will retain the same distribution shape. As a result, visually, the subsets may appear similar due to their adherence to the original distribution. Still, a closer examination of Figure 5.5b reveals a noticeable distinction in the density of the most common values, particularly in the range of 1 to 3 kPa, as their proportion alternates clearly between the three sets.

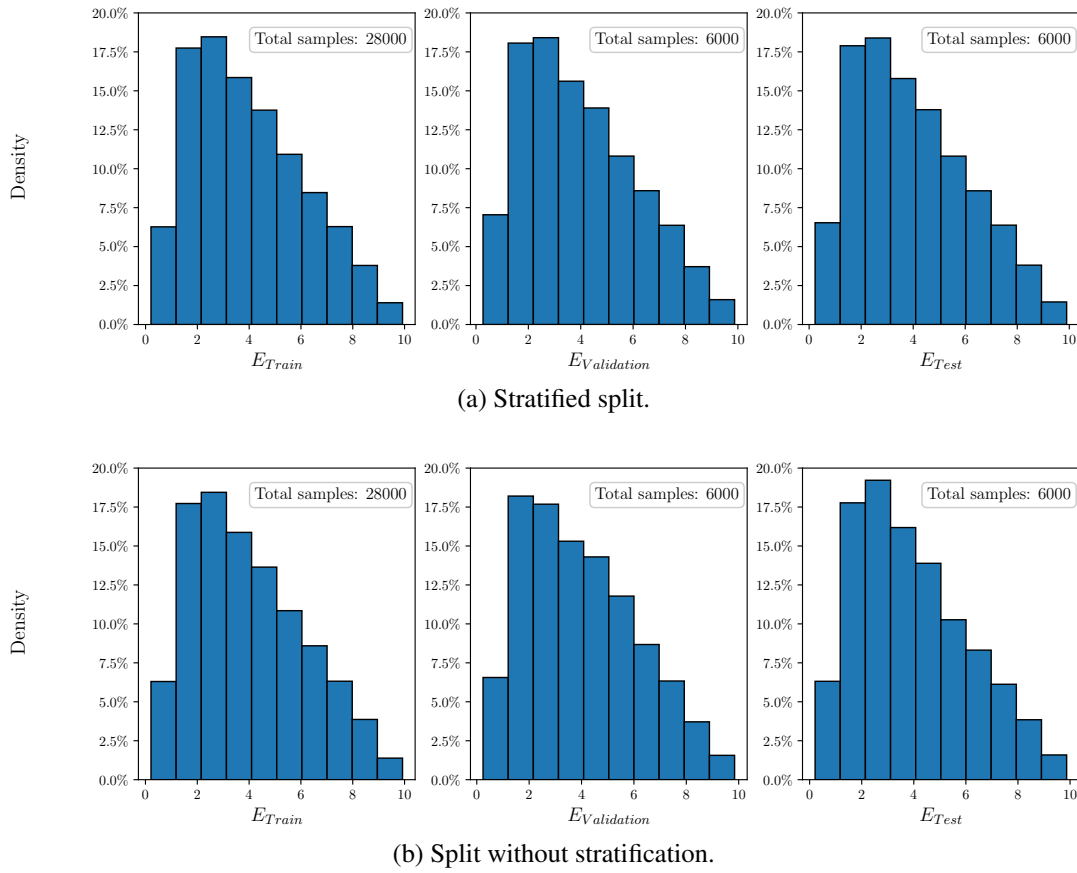


Figure 5.5: Distribution of the target variable over the different sets, with and without stratification.

The average test loss and error obtained for both splitting strategies are presented in Table 5.5. It is clear that the models already have a great performance on the testing set, using simply the

initial configuration. Even without stratification both loss and error present very low values, but it is also noticeable that just by applying a better splitting strategy, the results clearly improved. In addition, Figure 5.6 shows the error distribution in more detail, depicting the percentage of the test set curves that were predicted with an error below the thresholds of 2.5% and 10%. For the random state analysed, just by stratifying the data, there were 10.5% more curves being predicted with a relative error under 2.5%.

Table 5.5: Average test loss and error values for the model with and without stratification.

Splitting strategy	MSE Loss	$\bar{\epsilon}$
Stratified	$2.9 \cdot 10^{-3}$	1.8%
Without stratification	$4.2 \cdot 10^{-3}$	2.5%

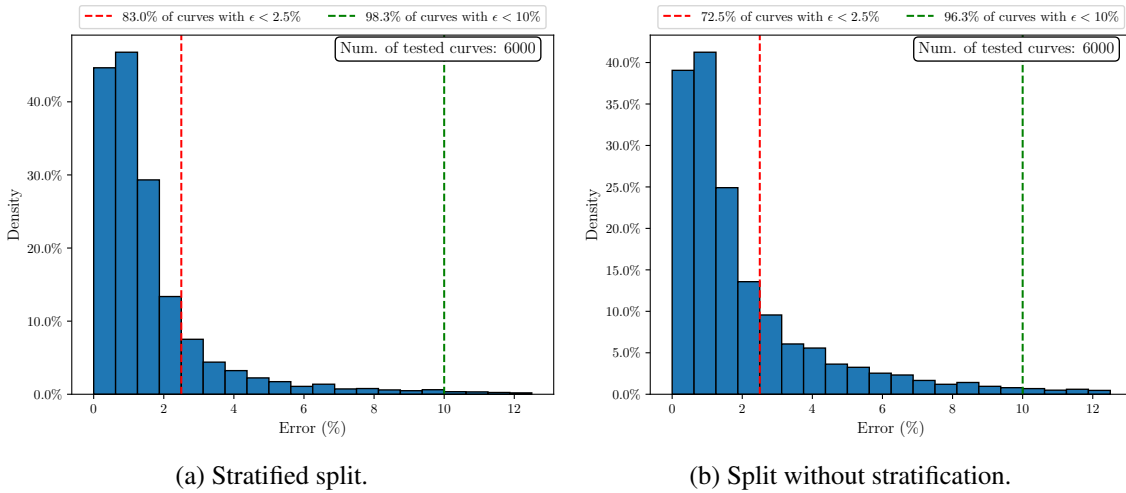


Figure 5.6: Distribution of the relative error for the testing samples.

Subsequently, an additional analysis was conducted to understand the influence of the split ratio on the MLP behaviour. For three training set proportions (60%, 70% and 80%), the model was trained in three random stratified splits for each training size and the average test loss (MSE) was computed. Validation and test ratios were always defined as the same value. This process was repeated for 20, 40 and 60 epochs, to gain deeper insights on how this factor interacts with the training duration.

The obtained results are graphically illustrated in Figure 5.7. We can see that using only 60% of data for training gives worst results, with the exception being when the ANN is trained for 40 epochs, where the 80% training split produced a higher loss value, which was not in agreement with the results of this split for 20 and 60 epochs. The original split (training set with 70% of the samples) presented a consistent behaviour for the three epoch values, despite displaying a higher loss than the 80% split for the lowest number of epochs. When the models were trained for 60 epochs, all showed good performance and relatively close loss values.

In conclusion, the analysis demonstrated the advantages of employing stratified data splitting during the model development. Three different split ratios were examined, and it was found that utilizing 70% of the data for training yielded the most favorable outcomes. Consequently, going forward, the proportion of the training set will be maintained at its initially selected value.

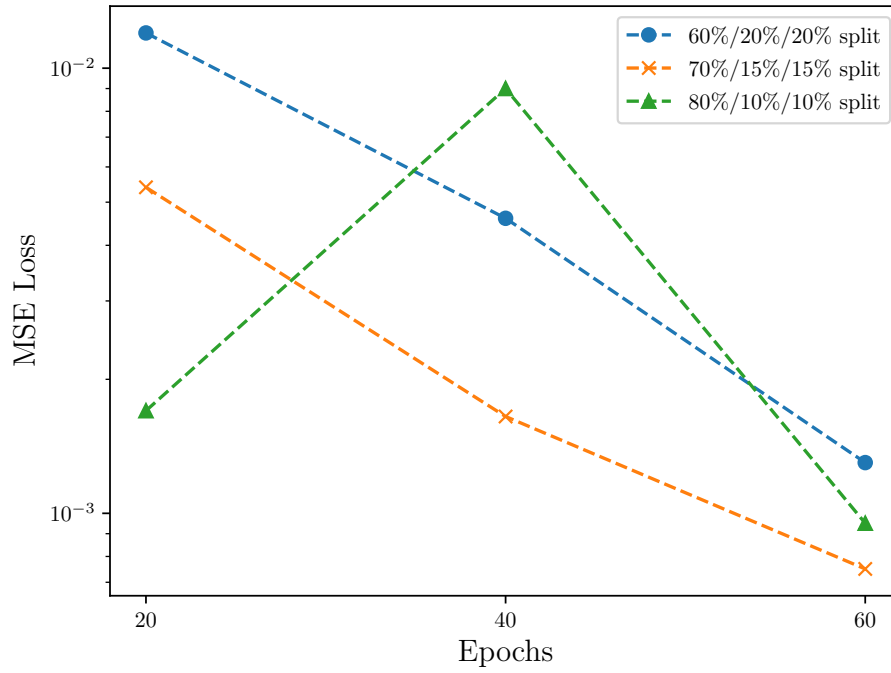


Figure 5.7: Test Loss (MSE) obtained for 3 different split ratios (Train/Test/Validation) and 20, 40 and 60 epochs. For each split proportion-epoch pair, 3 random splits were tested.

5.2.2 Activation and Loss Functions

In chapter 4, the key role of activation and loss functions in the development of NN models was discussed. Once the split ratio has been determined, it becomes crucial to carefully choose the activation function that introduces nonlinearity to the model and the appropriate loss function that quantifies the values the algorithm seeks to minimize.

Two activation functions were considered: ReLU and Leaky ReLU. For this last function, a constant of 0.01 was set, as the slope for the negative weighted inputs of each neuron. As for the loss, the MSE, MAE and Huber Loss functions were used. An evaluation of the various activation and loss functions was conducted by testing all possible configurations, while keeping the remaining hyperparameters constant, and equal to those presented for the base model.

The testing results for each configuration are displayed in Table 5.6. Besides the information of the loss and activation for each model, the MAPE ($\bar{\epsilon}$) is also presented, alongside the percentage of curves whose Young's moduli were predicted with an error below 2.5%, 5% and 10%.

Regarding the loss function, what becomes more noticeable is that the MAE has a worst accuracy compared to the other two, particularly in the MAPE value and in the percentage of curves with a very low relative error (2.5% and 5%). The Huber Loss stood out from the remaining in this last aspect, despite presenting similar $\bar{\epsilon}$ values compared to MSE. Applying the ReLU activation worked better than Leaky ReLU for the MSE and MAE loss functions, but the same did not happen for the Huber Loss.

Table 5.6: Error values for different loss and activation functions.

Loss function	Activation	$\bar{\epsilon}$	ϵ below 2.5%	ϵ below 5%	ϵ below 10%
MSE	ReLU	1.79%	82.95%	93.70%	98.27%
MSE	Leaky ReLU	2.79%	58.03%	84.45%	98.75%
MAE	ReLU	3.85%	21.00%	81.85%	98.37%
MAE	Leaky ReLU	3.38%	52.17%	77.38%	94.70%
Huber Loss	ReLU	2.27%	74.75%	92.5%	98.15%
Huber Loss	Leaky ReLU	2.06%	81.98%	91.52%	96.52%

Overall, the best performance was obtained using MSE and ReLU, which is nothing but the initial configuration defined for the Hertz MLP, whose error distribution was already shown in Figure 5.6a. However, when selecting Huber Loss and Leaky ReLU, the results were very similar for all of the presented evaluation metrics and the error distribution depicted in Figure 5.8 also confirms this aspect. Hence, this activation-loss function pair must also be accounted in the following hyperparameter tuning.

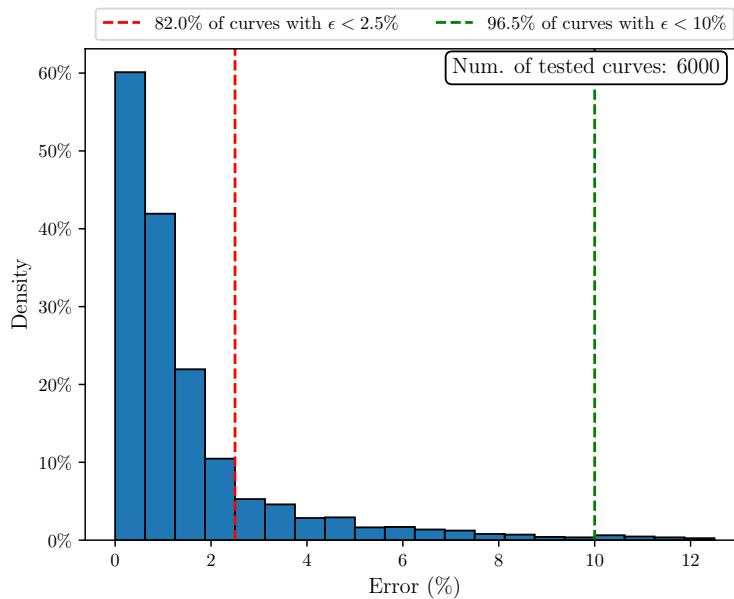


Figure 5.8: Error distribution when using the Huber Loss function and the Leaky ReLU activation, alongside the base model hyperparameters.

5.2.3 Optuna Implementation

If all the hyperparameters were manually tuned as previously done for the split ratio and the loss and activation functions, it would require a colossal amount of time to achieve their most suitable values. Thus, Optuna will be implemented to ease this process. It is a widely used Python package that runs with almost any ML framework, including PyTorch, and performs automatic hypertuning, by taking advantage of the performance history on the training process (1)(28). In Figure 5.9 there is a schematic representation of the workflow adopted for Optuna implementation, summarising its main steps, which will be explained in further detail.

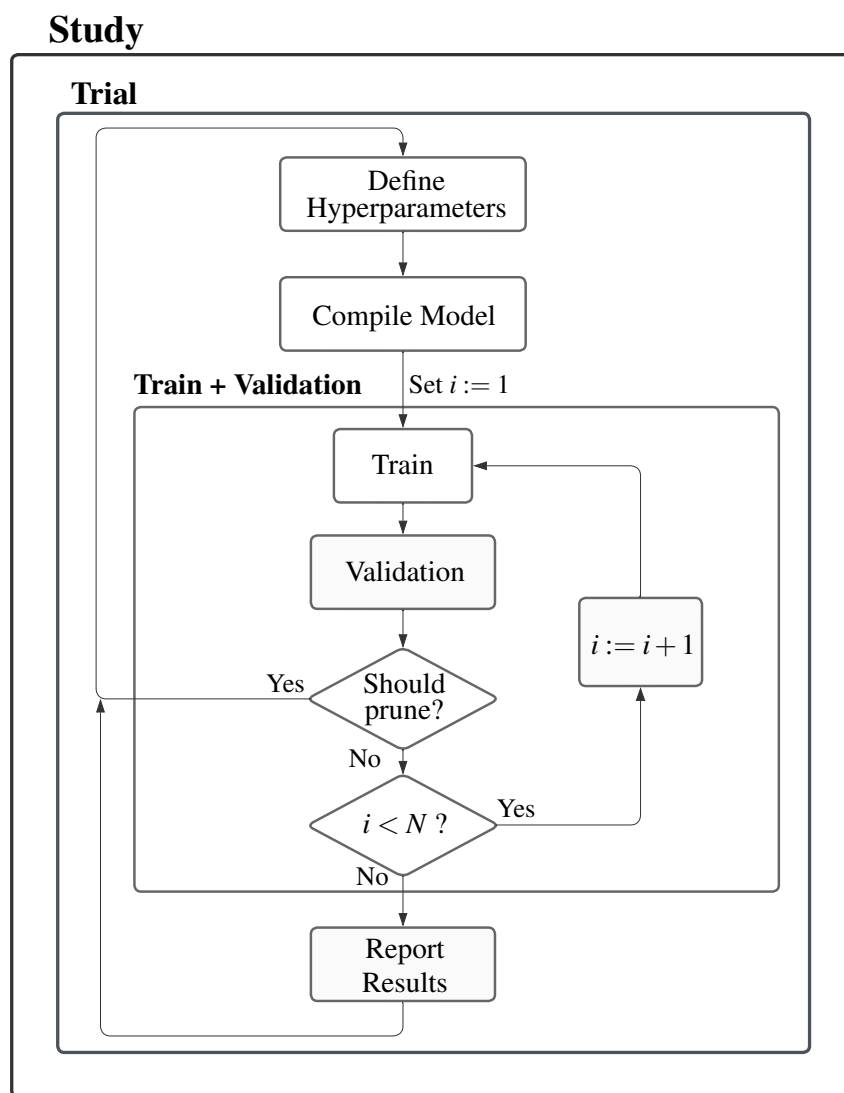


Figure 5.9: Optuna framework scheme with pruning, where N is the number of epochs.

Each hyperparameter analysis in Optuna is a "study", composed of a certain number of trials, with this number being defined by the user. Within each trial, it is first necessary to choose the

hyperparameters for the model, as one would do for a regular training process, but this time a range of values is provided, instead of only one value. Using the *suggest* function, it is possible to set up different types of hyperparameters, for instance: integers for the number of epochs or the batch size; floats for the learning rate and categorical values for the loss, activation function or optimization algorithm.

Before starting the analysis, an objective function has to be assigned for the "study", which in this case is to minimize the validation loss. After the algorithm has selected all the necessary hyperparameters from the ranges provided, the model must be compiled. Then, a regular training and validation cycle occurs for one epoch. At the end of each epoch, the mean validation loss is returned, and based on that value and the loss values obtained in previous trials, the algorithm decides if it is worth carrying on with the current trial or if it would be more efficient to start another trial with a different set of hyperparameters. This decision process is called pruning and can be defined as a method for discontinuing unpromising trials, leading to an important decrease in the computational cost for the hypertuning process. If the pruning algorithm states that the training cycle should go on, it is repeated until it gets pruned in a subsequent epoch or until the final number of epochs is reached. When this last scenario is verified, the algorithm reports the loss obtained for the corresponding trial, as well as the hyperparameters used. Then another trial starts, repeating this process until the maximum number of trials is reached.

Throughout this framework, the hyperparameter tuning was performed applying a *Median Pruner*, that prunes the trial if its best intermediate result at a given epoch is lower than the median of the intermediate results of previous trials, at the same epoch. To enhance reliability of the pruning algorithm, additional parameters can be set, as the minimum number of trials that must be ran before pruning gets activated, or a minimum number of epochs that the model must go through before it can be pruned. Algorithm 4 shows the pruning method proposed in (1), where besides the current *trial* and *epoch*, it takes as inputs the minimum resource m , a reducing factor λ and early stopping rate g .

Algorithm 4 Optuna's Pruner

```

procedure PRUNER(trial, epoch,  $m$ ,  $\lambda$ ,  $g$ )
   $rung \leftarrow \max(0, \log_{\lambda}(epoch/m) - g)$ 
  if  $epoch \neq m\lambda^{g+rung}$  then
    return false
  end if
   $value \leftarrow \text{get\_trial\_intermediate\_value}(trial, epoch)$ 
   $values \leftarrow \text{get\_all\_trials\_intermediate\_values}(epoch)$ 
   $k \leftarrow \text{len}(values)/\lambda$ 
   $top\_k\_values \leftarrow \text{top\_k}(values, k)$ 
  if  $top\_k\_values = \emptyset$  then
     $top\_k\_values \leftarrow \text{top\_k}(values, 1)$ 
  end if
  return  $value \notin top\_k\_values$ 
end procedure

```

Minimum resource m can be seen as a minimum number of epochs before the pruning check gets activated. The early stopping rate defines that pruning only happens if the intermediate value of the current trial is consistently lower than the intermediate values of the previous trials for a consecutive number of epochs specified by g .

The algorithm starts by calculating the variable *run*, that corresponds to how many times the current trial has escaped pruning. Then, it is checked if the current epoch is greater than the minimum number of epochs defined for pruning. If not, it returns *false* and the optimization goes on without pruning. Next, the intermediate values are obtained for the current trial and the previous ones (for the same epoch) and if the current intermediate value is lower than the k previous best intermediate values for the same epoch, the algorithm returns *true* and the trial gets pruned.

For the *Median Pruner* implementation, it was set that a minimum of 5 trials should be performed before the pruning was enabled in a study and that for each trial, at least 3 epochs should be completed.

5.2.3.1 Analysis 1 - Preliminary Model Evaluation

In the first Optuna implementation, only two studies will be performed, one considering the loss/activation functions combination of Huber/LeakyReLU and the other using MSE/ReLU. For each, a study of 100 trials was conducted, allowing Optuna to optimize an initial set of hyperparameters that are described in Table 5.7 (the remaining were maintained as in the base model).

Using the full initial dataset, the goal of this analysis is not only to have more information to decide on the loss and activation functions, but also to evaluate if there is any of the hyperparameters being studied that can already be fixed (or their initial ranges can be reduced), decreasing the complexity of future analysis.

Table 5.7: Hyperparameters for the Hertz MLP first analysis.

Hyperparameter	Type	Range
Learning Rate	Float	$[10^{-5}; 10^{-2}]$
Num. of Epochs	Integer	[10, 20, ..., 90, 100]
Optimizer	Categorical	[Adam, SGD]
Batch Size	Categorical	[16, 32, 64, 128]

For each study, 3 of the best trials were selected. They were not necessarily the best 3 of the study, since when one or more trials presented the same optimizer, number of epochs, batch size and a learning rate of the same magnitude, only one of them was considered, to ensure that we were not analysing points near the same local minima. Table 5.8 summarises the best trials for the Huber/LeakyReLU combination and Table 5.9 for MSE/ReLU.

The first thing that jumps out when analysing these tables is that the Adam optimizer was the preferred choice for all the best trials in both studies, thus it can be set as our final optimizer. The batch size only varied between 16 and 32, so our range can be reduced to only these two values.

Table 5.8: Best trials for the study with Leaky ReLU as the activation function.

Hyperparameter	Trials		
	1	2	3
Learning Rate	$5.5 \cdot 10^{-4}$	$6.7 \cdot 10^{-4}$	$3.4 \cdot 10^{-4}$
Num. of Epochs	80	50	40
Optimizer	Adam	Adam	Adam
Batch Size	32	16	32
Huber Loss	$1.57 \cdot 10^{-4}$	$1.84 \cdot 10^{-4}$	$1.85 \cdot 10^{-4}$

Table 5.9: Best trials for the study with ReLU as the activation function.

Hyperparameter	Trials		
	1	2	3
Learning Rate	$1.6 \cdot 10^{-4}$	$6.4 \cdot 10^{-4}$	$2.0 \cdot 10^{-4}$
Num. of Epochs	100	90	40
Optimizer	Adam	Adam	Adam
Batch Size	16	16	16
MSE Loss	$1.78 \cdot 10^{-4}$	$2.63 \cdot 10^{-4}$	$2.70 \cdot 10^{-4}$

Then, two models with the hyperparameters of the best trial for each study were trained and evaluated on the testing set. As the model error gets lower by each optimization step, the percentage of curves with an error below 1% becomes more relevant, so it is already presented in Figure 5.10. The graphs in this figure show the test results for both models and this time there was a clear improvement when using the Huber Loss and Leaky ReLU, compared to the other setting. So these are defined as our loss and activation functions, respectively.

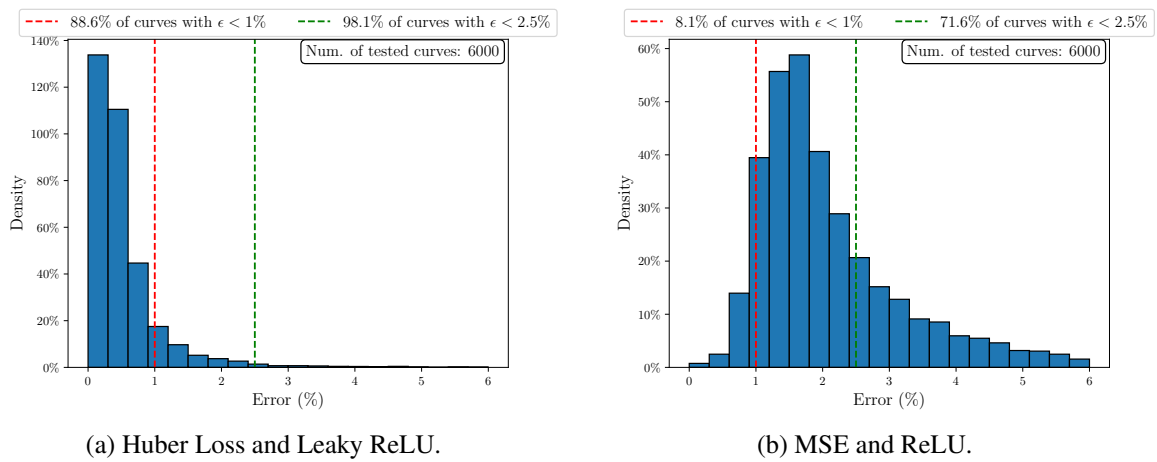


Figure 5.10: Relative error distribution for the 2 models analysed.

5.2.3.2 Analysis 2 - Evaluate Overall Architecture

For this second analysis, Optuna was allowed to optimize the overall architecture of the MLP, with exception for the hyperparameters that have been defined in Analysis 1 (optimizer, loss and activation functions). This approach aimed at determining the hyperparameters that were having a greater impact in constraining the model, providing the necessary insights if further fine-tuning was needed or some more variables could already be settled.

This time the width and depth of the NN were also included in the optimization process. As the previous models were already providing a low loss for our target variable, it was only allowed to increase its depth in one layer, while the maximum number of nodes was set as the width of base model's first hidden layer: 256. The hyperparameters being studied and their enabled ranges are shown in Table 5.10.

Table 5.10: Hyperparameters being optimized in the second analysis of the Hertz MLP.

Hyperparameter	Type	Range
Learning Rate	Float	$[10^{-5}; 10^{-2}]$
Num. of Epochs	Integer	[10, 20, ..., 90, 100]
Batch Size	Categorical	[16, 32]
Num. of Hidden Layers (HL)	Integer	[2, 3]
Nodes in each HL	Categorical	[16, 32, 64, 128, 256]

In total, 5 studies of 100 trials each were performed. To decrease the computational cost of the optimization, 4 of those studies used only a subset of 10 000 instances, instead of the initial 40 000. To assess if the suggested hyperparameters would be very different if the initial dataset was used in this optimization, a fifth study was done, with the complete dataset. The results obtained for each study can be found in Table 5.11.

Table 5.11: Best trials in each of the 5 studies performed in Analysis 2.

Hyperparameter	Study				
	1	2	3	4	5
Learning Rate	$3.59 \cdot 10^{-4}$	$3.62 \cdot 10^{-4}$	$10.5 \cdot 10^{-4}$	$6.47 \cdot 10^{-4}$	$3.56 \cdot 10^{-4}$
Num. of Epochs	90	100	100	90	90
Batch Size	16	16	16	16	16
Num. of HL	3	3	3	3	3
Nodes HL 1	64	128	128	256	128
Nodes HL 2	64	64	32	128	128
Nodes HL 3	16	64	64	128	64
Huber Loss	$7.96 \cdot 10^{-4}$	$7.01 \cdot 10^{-4}$	$7.94 \cdot 10^{-4}$	$6.06 \cdot 10^{-4}$	$1.70 \cdot 10^{-4}$

Many conclusions can be withdrawn from these results. For the best trial in every study, a batch size of 16 was the preferred choice and in Figure 5.11 it is shown (for studies 1 to 4) that the lowest losses were achieved with that batch size, in addition to a lower mean loss across all trials. The number of epochs was always near the maximum allowed, which could indicate that this parameter is limiting the model's performance. However, further increasing this range could cause overfitting and lead to a worst behaviour when doing predictions on the experimental dataset, so its maximum value will stay the same, which can also be supported by the fact that in 3 out the 5 studies, its maximum value wasn't reached.

A hyperparameter that could see its allowed range decrease is the learning rate, since it was always between 10^{-4} and 10^{-3} . Another common factor across all studies was that the best results were produced with 3 hidden layers. From the number of nodes in each layer, the main aspect to take away is that the first layer is commonly the largest, and the ANN width tends to decrease over the sequential layers.

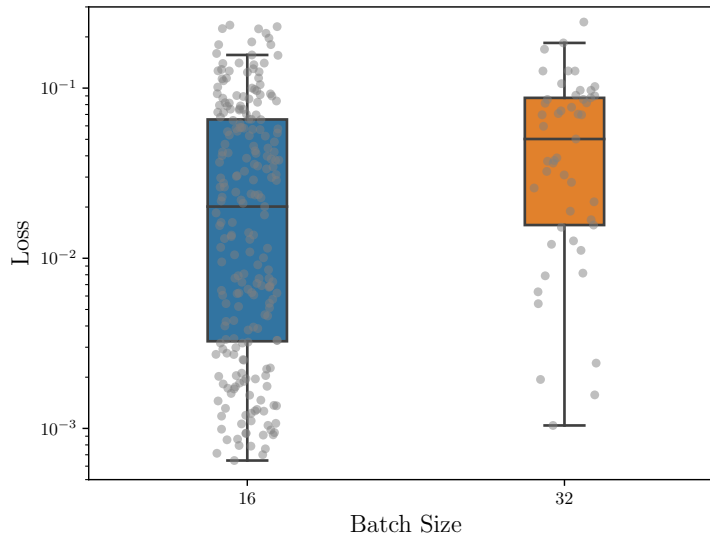


Figure 5.11: MSE loss obtained for a batch size of 16 and 32, over several trials of this analysis.

To summarise this study, the number of hidden layers will be fixed at 3 and the batch size as 16. The learning rate range will decrease towards the middle values in the current range and the number of epochs will see its range decreased towards its current maximum value.

5.2.3.3 Analysis 3 - Defining Model's Width

Analysis 3 focuses mainly in defining the number of neurons for all three hidden layers. Learning rate and number of epochs are the other hyperparameters that the algorithm is able to optimize in this study, but they've seen their allowed interval reduced compared to the last analysis, as shown in Table 5.12. The range of neurons for each layer has also been adapted, considering the results in Analysis 2.

Table 5.12: Hyperparameters being optimized in the Analysis 3.

Hyperparameter	Type	Range
Learning Rate	Float	$[8 \cdot 10^{-5}; 2 \cdot 10^{-3}]$
Num. of Epochs	Integer	[60, ..., 90, 100]
Nodes in HL1	Categorical	[64, 128, 256]
Nodes in HL2	Categorical	[32, 64, 128, 256]
Nodes in HL3	Categorical	[16, 32, 64, 128]

Once again, 5 studies were performed, 4 of them for a reduced set composed of 10 000 curves and other (Study 5) for the full dataset, with the results being presented in Table 5.13.

Table 5.13: Best trials for each study in Analysis 3.

Hyperparameter	Study				
	1	2	3	4	5
Learning Rate	$8.00 \cdot 10^{-4}$	$3.73 \cdot 10^{-4}$	$3.38 \cdot 10^{-4}$	$4.90 \cdot 10^{-4}$	$1.46 \cdot 10^{-4}$
Num. of Epochs	100	80	100	100	80
Nodes HL 1	256	256	256	128	256
Nodes HL 2	256	256	256	256	128
Nodes HL 3	16	32	16	32	32
Huber Loss	$7.33 \cdot 10^{-4}$	$3.63 \cdot 10^{-4}$	$5.49 \cdot 10^{-4}$	$4.66 \cdot 10^{-4}$	$3.30 \cdot 10^{-4}$

In most studies, for their best trials, the number of neurons of the first two hidden layers was set as 256, so this can be defined as our final choice for those layers.

For the final layer, 16 and 32 were the preferred values. Figure 5.12 shows a box plot of the loss obtained for the trials across all studies where the third layer had a depth of 16 or 32. Losses higher than 0.1 were discarded as they are likely to be associated with early stage pruning. Despite the loss median being slightly higher for 16 neurons, using 32 neurons in this layer generated at least 4 trials with a lower loss than the other configuration, hence being more likely to reach a better local minimum. Taking this into account, 32 neurons will be the choice for our final layer.

As for the number of epochs, there was also some division between 80 and 100 epochs. To try and prevent overfitting, as mentioned in the last analysis, the lower number of epochs will be the final choice, so it will be set as 80.

The learning rate did not reach the extreme values of the proposed interval, which legitimises its reduction from the previous analysis to the current one.

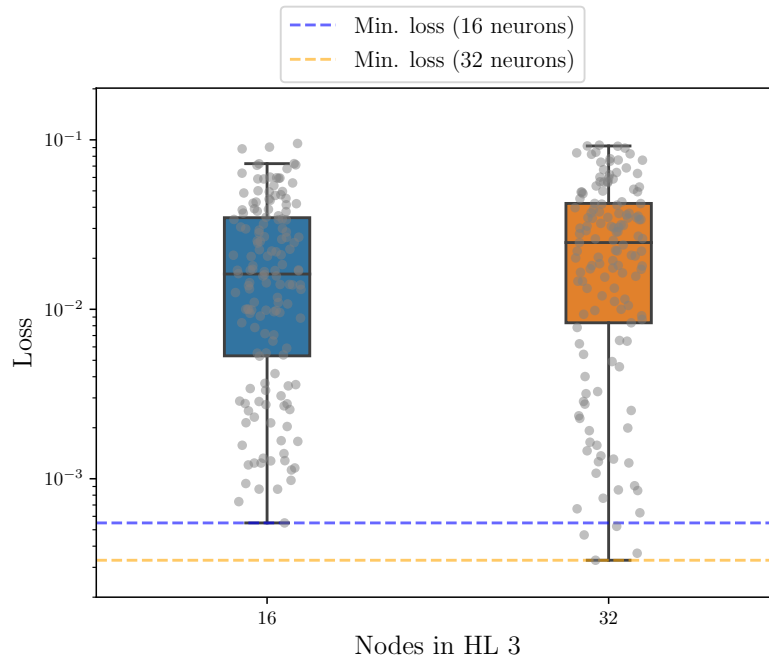


Figure 5.12: MSE loss obtained from using 16 or 32 nodes in the last hidden layer, across the trials for all studies of analysis 3.

5.2.3.4 Analysis 4 - Setting the Learning Rate

Now that all the remaining hyperparameters have been sequentially defined through the previous analyses, only the learning rate is left. We've seen throughout this section that the MLP being studied is very sensitive to learning rate variations, so that changes in the magnitude of 10^{-4} and even lower have a great impact on the loss obtained.

As only one variable was being studied, this time a simpler analysis was conducted, with three studies of 100 trials each, using a reduced dataset of 10 000 instances. The allowed range for the learning rate is the same as in Analysis 3 and it is presented in Table 5.14.

Table 5.14: Allowed range for the Learning Rate in Analysis 4.

Hyperparameter	Type	Range
Learning Rate	Float	$[8 \cdot 10^{-5}; 2 \cdot 10^{-3}]$

The best trials for each study are presented in Table 5.15. Once again, the lowest losses were obtained for learning rates that are closer to the minimum value of the defined interval. In Figure 5.13 is depicted the distribution of the losses in order to the learning rate for all the trials in the three studies. Only losses lower than 0.01 are represented, since most trials with a higher loss that were pruned at some point.

We can see there is a slight correlation between the increase of the learning rate and a loss increase, but mainly for rates higher than $3 \cdot 10^{-4}$. Other than that, it confirms the high sensitivity of the model performance to slight variations in the hyperparameter being studied. To sum up this last analysis, the learning rate chosen was the one that resulted in the best trial of Study 2.

Table 5.15: Best trials for the 3 studies in Analysis 4.

Hyperparameter	Study		
	1	2	3
Learning Rate	$0.994 \cdot 10^{-4}$	$2.296 \cdot 10^{-4}$	$7.147 \cdot 10^{-4}$
Huber Loss	$4.68 \cdot 10^{-4}$	$3.92 \cdot 10^{-4}$	$8.11 \cdot 10^{-4}$

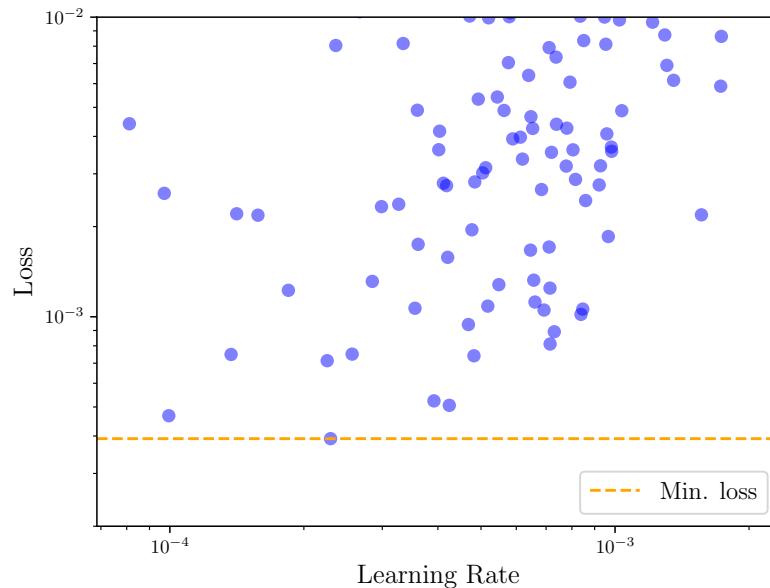


Figure 5.13: Relation between Loss and Learning Rate for different trials over the three studies .

5.2.4 Evaluating the Models on Synthetic Data

After optimizing all the hyperparameters with Optuna, it is important to understand how each analysis improved the model, or to check if any improvements at all were verified from one analysis to another.

Thus, for each analysis, a model will be selected with the hyperparameters that led to the lowest loss. The only exception is Analysis 3, where very similar loss values were generated for the best trials in Studies 2 and 5 (slightly lower for Study 5), but since Study 2 was trained on much less instances, it will be the preferred choice. The goal is to test each model on the original testing set and infer the more successful optimization steps.

In total, 5 models will be tested: one for each analysis (that will be named Model 1 for Analysis 1 and from there on) and the base model initially presented. To aid in the comprehension of the testing performance for each model, their hyperparameters are summarised in Tables 5.16.

Table 5.16: Hyperparameters of the final Hertz MLPs. All used Adam as the optimization algorithm, while Leaky ReLU and Huber Loss were applied in all, with the exception of the base model (ReLU and MSE).

Model	HL	Nodes HL 1	Nodes HL 2	Nodes HL 3	Epochs	LR	Batch
Base Model	2	256	64	-	20	10^{-3}	32
Model 1	2	256	64	-	80	$5.50 \cdot 10^{-4}$	32
Model 2	3	128	128	64	90	$3.56 \cdot 10^{-4}$	16
Model 3	3	256	256	32	80	$3.73 \cdot 10^{-4}$	16
Model 4	3	256	256	32	80	$2.30 \cdot 10^{-4}$	16

After having undergone training and validation in the initial dataset with the same split ratio (70% for training), the testing set was used to evaluate these models, which performed according to the results in Table 5.17. The column " $\Delta\bar{\epsilon}$ " represents the difference in MAPE in relation to the base model. Figure 5.14 details the relative error distribution for the models that presented the worst and best performance on the synthetic test set.

Table 5.17: Performance of the models on the synthetic test set.

Model	Loss	$\epsilon < 10\%$	$\epsilon < 5\%$	$\epsilon < 2.5\%$	$\epsilon < 1\%$	$\bar{\epsilon}$	$\Delta\bar{\epsilon}$
Base Model	$28.75 \cdot 10^{-4}$	98.27%	93.70%	82.95%	45.27%	1.79%	-
Model 1	$2.10 \cdot 10^{-4}$	99.83%	99.5%	98.10%	88.65%	0.55%	-1.24%
Model 2	$2.67 \cdot 10^{-4}$	99.88%	98.88%	93.63%	70.65%	0.89%	-0.9%
Model 3	$9.62 \cdot 10^{-4}$	98.30%	91.47%	74.85%	49.35%	1.88%	+0.09%
Model 4	$17.33 \cdot 10^{-4}$	97.97%	89.60%	67.10%	32.12%	2.40%	+0.61%

The first thing that comes to sight when looking at the results is that models 3 and 4 performed worst than the base model on the test set. Based on that, one could tend to say that both these models could be eliminated and only Models 1 and 2, that improved in relation to the initial model, should be proposed as the final MLPs to be tested on the experimental set.

Two hypothesis can be formed based on these results. The first is that Models 1 and 2 were trained in a way that they were able to recognize the patterns in training data and to generalize to the testing set, opposing to Models 3 and 4. The second is that Models 1 and 2 simply memorized the synthetic data patterns (they were optimized with the full initial dataset, unlike Models 3 and 4), performing well on this type of data, whether in the training, validation or test sets, but won't be able to generalize to real applications, thus presenting a bad behaviour on the experimental set.

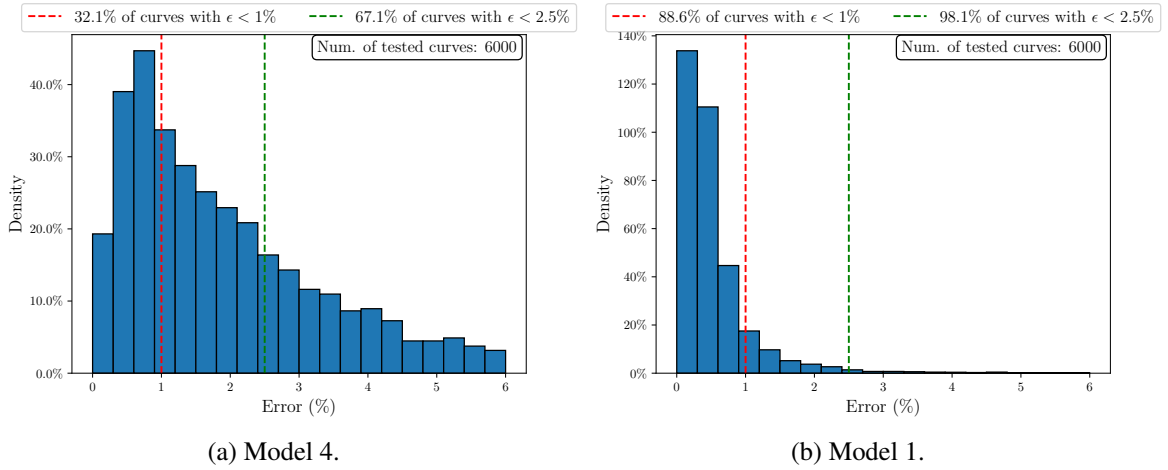


Figure 5.14: Relative error distribution for the models that behaved worst and best on the synthetic testing set, respectively.

Of course this testing on the synthetic data is only an intermediate step in the global view of this framework, so all these models will also be tested with the experimental F-I curves, to then conclude which hypothesis was correct.

5.2.5 Hertz MLP Concluding Remarks

Throughout the development of the Hertz MLP, several ML aspects have been considered. After defining a base model, it was demonstrated the importance of stratifying the data when doing the train-validation-test split and an appropriate split ratio was chosen. Then, it was verified which were the better options for the loss and activation functions, from a naive approach.

The implementation of Optuna was important to optimize the MLP hyperparameters in a more efficient manner and sequentially reduce their target intervals until eventually reaching their final value. In the end, the models resulting from the first two Optuna analysis performed better than the ones which suffered greater refinement.

Throughout this section the MAPE and the relative error were commonly used as evaluation metrics and despite being easy to understand what is the impact of a certain relative error in a Young's modulus value, it is not that intuitive to realize what are its consequences in an AFM F-I approach curve. To bridge this gap, the graphs presented in Figure 5.15 depict a comparison between the actual synthetic curves and the curves that would be produced with the predicted Young's modulus for that actual curve, obtained for different models developed along this section, with relative errors of 2%, 10% and higher than 20%.

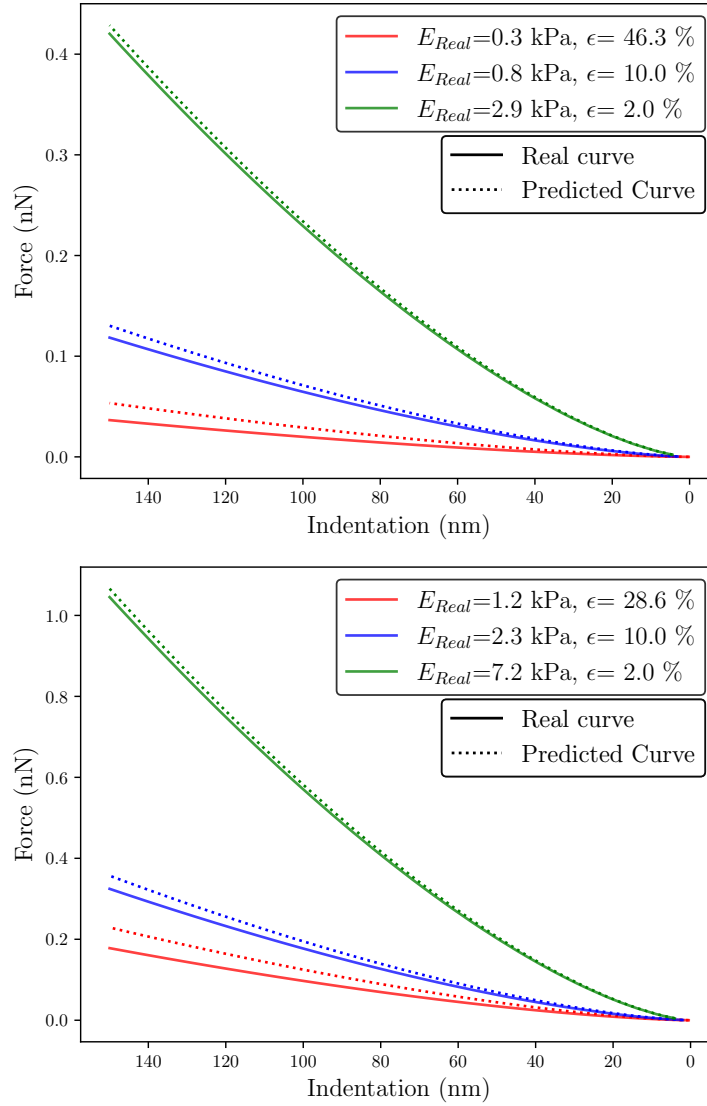


Figure 5.15: Actual and predicted synthetic curves for different Young's moduli with relative errors near 2%, 10% and greater than 20%.

5.3 Development of the JKR MLP

The second model developed in this framework has also the architecture of a Multilayer Perceptron, fitting also in the category of the Fully Connected Neural Networks.

Despite using the same 40 000 initially generated instances, it differs from the Hertz MLP in the sense that it works with the retraction curves of the dataset and its goal is to predict two target variables, instead of only one, so it is expected that this model requires higher complexity than the previous one. With this in mind, the base model configuration will also be more robust. We'll start off by creating the model with 3 hidden layers (HL), each having 256, 64 and 32 nodes, respectively. It will also be initially trained for longer, by setting the number of epochs to 50. The learning rate was lowered to $5 \cdot 10^{-4}$, while the batch size was maintained at 32.

The remaining initial variables were kept the same as in the base Hertz MLP: ReLU as the activation function, loss computed as the MSE, Adam set as the optimizer and 70% of the data were assigned to the training set, while the remaining were equally split for validation and testing.

Table 5.18: Hyperparameters of the baseline JKR MLP model.

HL	Nodes HL 1	Nodes HL 2	Nodes HL 3	Epochs	Learning Rate	Batch Size
3	256	64	32	50	$5 \cdot 10^{-4}$	32

Table 5.19: Initial activation and loss functions, optimization algorithm and split ratios for the JKR MLP.

Activation	Loss	Optimizer	Data split		
			Train	Validation	Test
ReLU	MSE	Adam	70%	15%	15%

Mean Absolute Percentage Error (MAPE) or simply $\bar{\epsilon}$ will remain a common metric in the evaluation of this model, and since now there are two output variables, a frequent distinction will be made between them, by presenting ϵ_E and ϵ_γ as the relative error for the Young's modulus and the adhesion energy, respectively.

5.3.1 Stratification and Split Ratio

The importance of stratifying the data over the three sets increases with the increase of target variables, so it must be studied again for the new model, where the target parameters were split into 30 bins to allow stratification. As the difference in the distribution of the variable E , with and without stratification, has been shown in the Hertz MLP development, here in Figure 5.16, only the allocation of γ in each set is presented. Having an uniform distribution, it is easier to graphically identify the differences between the two splitting strategies.

By selecting the hyperparameters of the base model and changing only the split strategy, the relative error results obtained for E and γ are presented in Table 5.20. Once again, it is noticeable that stratifying the data produced better results in predicting both material parameters. As it can be seen in Figure 5.17, by using stratification, there was an increase of about 32% in the curves where E was predicted with an error below 2.5% and an increase close to 10% for the energy of adhesion, regarding the same evaluation metric.

Overall, the base JKR MLP configuration produced very good results, even for a split without stratifying data. However, it is certain that there are other sets of hyperparameters that allow improving the results, even using less computational resources.

The split ratio was also studied for this model, using the same three training set proportions than in the Hertz MLP (60%, 70% and 80%), and training these models for 25, 50 and 75 epochs. To obtain more reliable outcomes, each model was trained with 3 random splits for each epoch and split proportion combination.

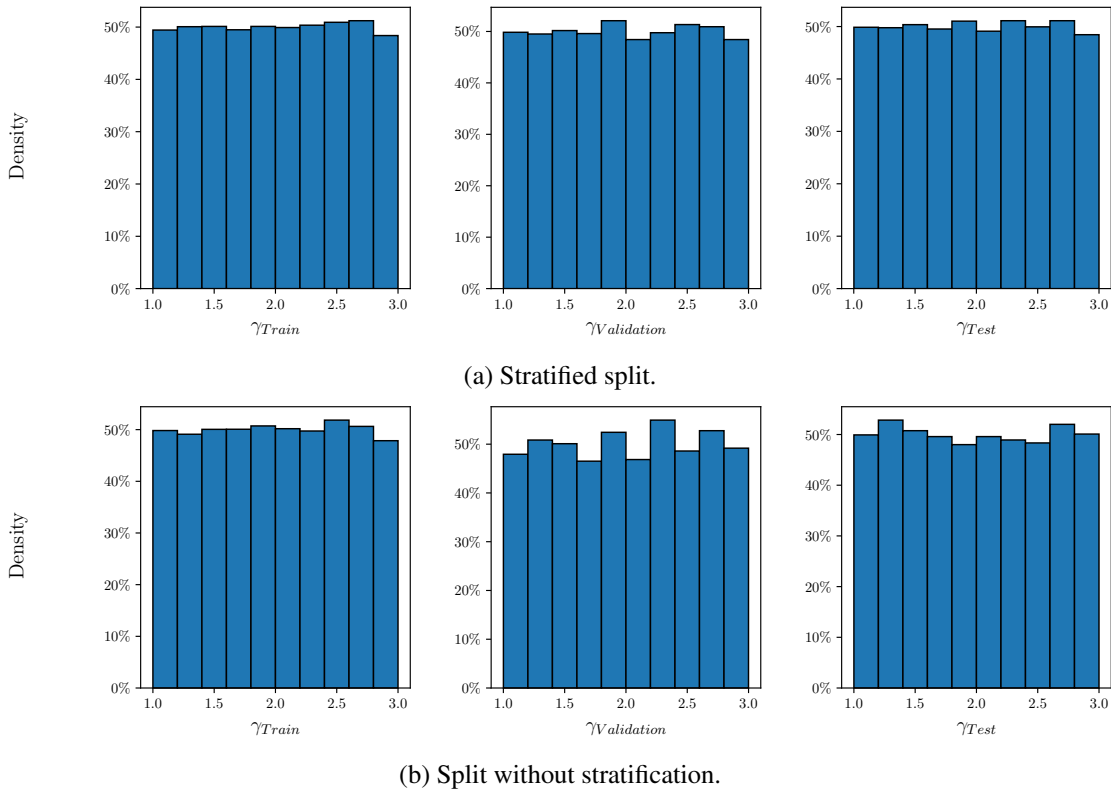
Figure 5.16: Distribution of the variable γ over the different sets, with and without stratification.

Table 5.20: Average test error values for both target variables, with and without stratification.

Splitting strategy	$\bar{\epsilon}_E$	$\bar{\epsilon}_\gamma$
Stratified	1.26%	1.87%
Without stratification	2.94%	2.38%

To get a better insight on the predictions of each variable, Figure 5.18 shows the MSE loss for each one separately, instead of combining them into a single value. When 80% of the data is used for training, the model presents worst performance for higher epochs and for both material parameters, which can indicate that it is overfitting, so it is not an appropriate split ratio. Regarding the other two, their behaviour is very similar for the loss related to the Young's modulus, with a slight advantage for the 60% split, but the split ratio from the initial configuration performs much better in adhesion energy prediction, besides presenting the best overall accuracy.

To sum up, the benefits of stratification have been showed once again and different split ratios were tested for a different number of epochs. The split where 70% of the data was attributed to the training set presented the best performance and will be used in the following optimization of the JKR MLP.

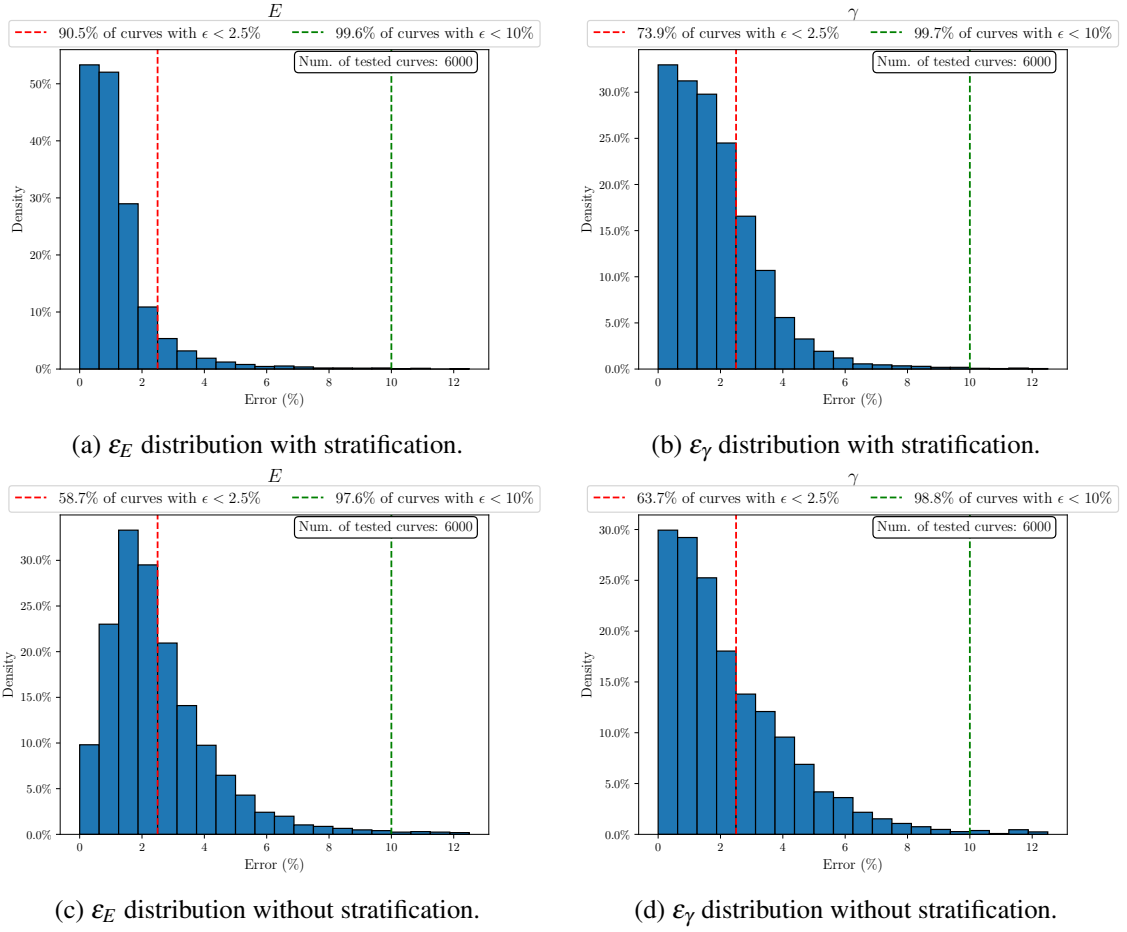


Figure 5.17: Relative error distribution for the material parameters in the testing set, with and without data stratification.

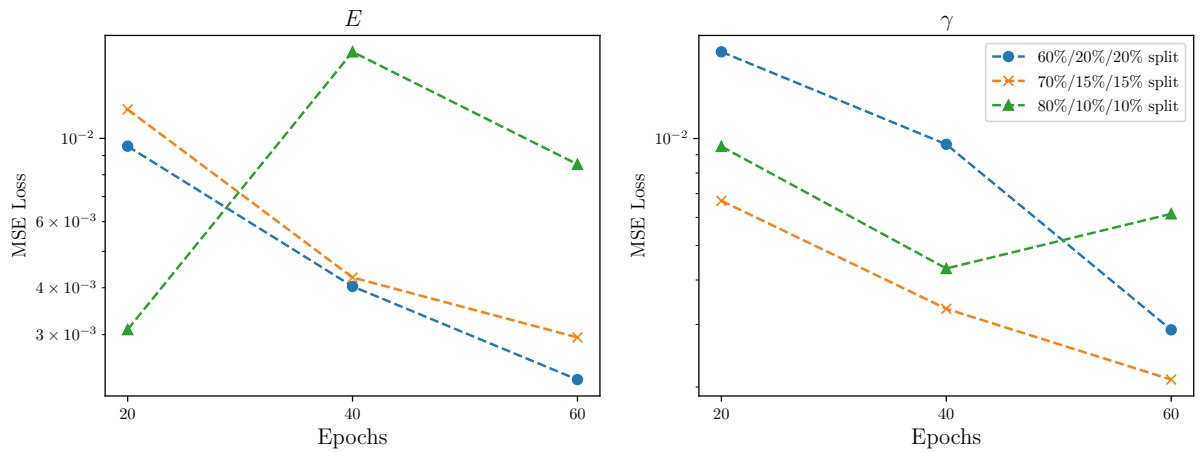


Figure 5.18: MSE loss obtained in the testing set with different split ratios and number of epochs, for both material parameters (E at the left and γ at the right).

5.3.2 Activation and Loss Functions

After having the split ratio determined, we can move on to the choice of the activation and loss functions. For the activation there is no change in the functions to be studied compared to the Hertz MLP: ReLU and Leaky ReLU (again with a constant of 0.01) will be the ones considered.

The greater difference is related to the loss functions. As now there are two continuous variables, if they had two very different scales, using regular functions as the MSE or MAE losses could in some cases result in similar loss values for both variables, but that would then be translated into very different relative errors. It is not expected to be the case of the current framework, as both material parameters do not differ very much in their order of magnitude, but this component must be inferred anyways. Thus, MSE and Huber Loss will be compared to a custom loss function based on the MAPE (there is no corresponding function already implemented in PyTorch), so that the loss is calculated directly on the relative error. All the 6 configurations were trained and tested on the initial dataset, generating the results presented in Table 5.21.

Table 5.21: Error values for different loss and activation functions.

Loss function	Activation	$\bar{\epsilon}_E$	$\epsilon_E < 10\%$	$\epsilon_E < 5\%$	$\bar{\epsilon}_\gamma$	$\epsilon_\gamma < 10\%$	$\epsilon_\gamma < 5\%$
MSE	ReLU	1.26%	99.62%	97.82%	1.87%	99.65%	96.43%
MSE	Leaky ReLU	2.90%	94.85%	83.33%	1.93%	98.37%	91.77%
MAPE	ReLU	1.82%	99.13%	95.83%	8.50%	69.50%	22.25%
MAPE	Leaky ReLU	1.61%	99.07%	95.78%	2.44%	99.45%	90.70%
Huber Loss	ReLU	4.46%	88.85%	68.08%	3.89%	96.15%	74.25%
Huber Loss	Leaky ReLU	1.99%	99.5%	97.75%	5.74%	90.45%	46.72%

The option that allowed more accurate and consistent results for both parameters was using MSE and ReLU. Regarding the loss functions, using the MAPE or the Huber Loss generated higher variation in the error between the use of the ReLU and Leaky ReLU. Overall, MAPE with ReLU and the Huber Loss with both activation functions presented the worst behaviour.

MSE provided more consistent predictions, regardless of the activation function. Despite MSE and ReLU showing the best results, the combination of MAPE and Leaky ReLU also had a quite good performance and MAPE has the advantage of not depending on the outputs scale. Hence, these two best combinations were evaluated over 4 additional unseen test folds, to see if their performance was consistent with the previous results. These folds had the same dimension as the test set, a triangular distribution for E and uniform for γ . Figure 5.19 shows the outcomes of this new evaluation (considering also the original test set). It is possible to conclude that both configurations had a consistent performance and that MSE and ReLU presented the best results for the two properties, so they'll be set as our loss and activation functions.

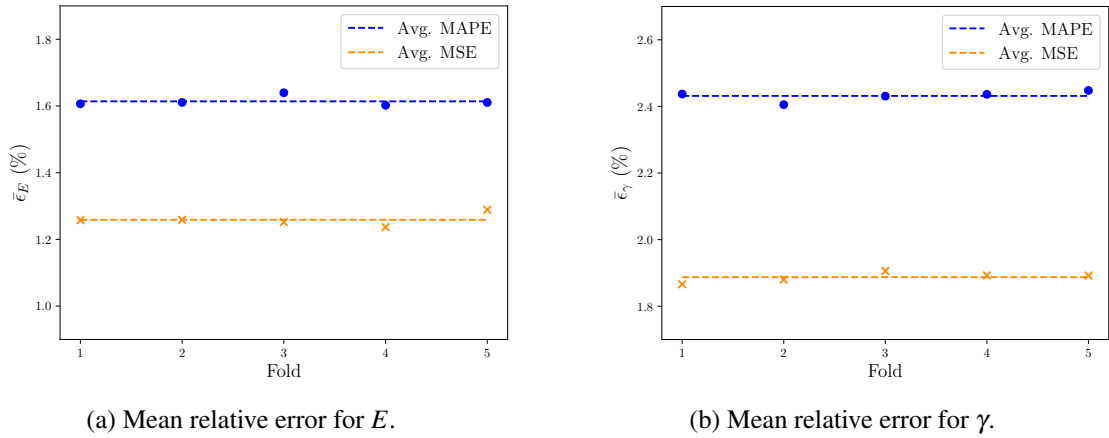


Figure 5.19: Average relative error obtained for the 5 test folds (initial test set and 4 additional folds), using MAPE and MSE as loss functions.

5.3.3 Optuna Implementation

Optuna's implementation will be similar to the one previously presented for the Hertz MLP. The loss presented throughout the analysis will be the average MSE loss for both parameters, which is not the perfect final evaluation metric (and it won't be used as so), but it is suitable to compare the models within each analysis.

The *Median Pruner* was also implemented on the same principles, ensuring that at least 5 trials got completed before the pruning algorithm got activated and within each trial the first 3 epochs weren't subjected to being pruned.

5.3.3.1 Analysis 1 - Preliminary Model Evaluation

Since the loss and activation functions from our base model were already defined as the ones to be further evaluated, the foundation for this first analysis will be our initial model. The initial objective is just to reduce the broader initial range of allowed values for the hyperparameters being studied, and if possible define already one of the them.

Hence, the model's width and depth won't be evaluated here and we'll focus on the learning rate, number of epochs, optimizer and batch size. Since the JKR MLP has two target variables, the maximum allowed number of epochs will be increased and the minimum learning rate will be decreased. The intervals for each hyperparameter in this analysis are shown in Table 5.22.

Table 5.22: Hyperparameters for the JKR MLP first analysis.

Hyperparameter	Type	Range
Learning Rate	Float	$[10^{-6}; 10^{-2}]$
Num. of Epochs	Integer	[10, 20, ..., 140, 150]
Optimizer	Categorical	[Adam, SGD]
Batch Size	Categorical	[16, 32, 64, 128]

Since this is a preliminary analysis, only 2 studies were conducted, with the full initial dataset. As the goal is not to find the final value for one hyperparameter, if two trials present the 3 integer or categorical variables with the same value between them, only one of them will be selected, like it was already done in the Hertz MLP. Three of the best trials for each study are presented in Table 5.23.

Table 5.23: Best trials for the studies in the first analysis of the JKR MLP.

Hyperparameter	Study 1			Study 2		
	T1	T2	T3	T1	T2	T3
Learning Rate	$2.82 \cdot 10^{-3}$	$4.15 \cdot 10^{-4}$	$1.89 \cdot 10^{-3}$	$5.09 \cdot 10^{-4}$	$6.73 \cdot 10^{-4}$	$1.07 \cdot 10^{-3}$
Num. of Epochs	140	120	80	100	110	130
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Batch Size	32	16	32	16	16	16
Loss	$2.46 \cdot 10^{-3}$	$3.23 \cdot 10^{-3}$	$3.88 \cdot 10^{-3}$	$0.97 \cdot 10^{-3}$	$1.11 \cdot 10^{-3}$	$1.40 \cdot 10^{-3}$

The two studies demonstrated that there is one hyperparameter that can also be established: the optimizer. Throughout all the trials, SGD consistently presented a much worst performance than Adam, so this last optimizer will remain constant in future analysis.

Regarding the other hyperparameters, it becomes clear that their range can be reduced. From now on, the batch size will only vary between 16 and 32, the minimum number of epochs will be set as 60 and the minimum learning rate will be raised, as it did not get even close to 10^{-6} in all good trials.

5.3.3.2 Analysis 2 - Evaluate Overall Architecture

Now that the optimizer has been defined and the other hyperparameters have seen their allowed intervals reduced, the number of layers and neurons in each layer will be added to the optimization process. We'll start by assigning the same range for the width in each layer, while enabling the addition of a new hidden layer. 5 studies were performed, all of them using a reduced dataset of 10 000 instances. The allowed range for each hyperparameter is presented in Table 5.24.

Table 5.24: Hyperparameters being optimized in the second analysis of the JKR MLP.

Hyperparameter	Type	Range
Learning Rate	Float	$[10^{-5}; 10^{-2}]$
Num. of Epochs	Integer	[60, 70, ..., 140, 150]
Batch Size	Categorical	[16, 32]
Num. of Hidden Layers (HL)	Integer	[3, 4]
Nodes in each HL	Categorical	[16, 32, 64, 128, 256]

The hyperparameters that resulted in the best trials for each study are summarised in Table 5.25.

Table 5.25: Best trials in each of the 5 studies performed in Analysis 2.

Hyperparameter	Study				
	1	2	3	4	5
Learning Rate	$8.38 \cdot 10^{-4}$	$1.07 \cdot 10^{-3}$	$5.94 \cdot 10^{-4}$	$1.03 \cdot 10^{-3}$	$2.00 \cdot 10^{-3}$
Num. of Epochs	110	130	120	110	100
Batch Size	16	16	32	16	16
Num. of HL	4	3	3	3	3
Nodes HL 1	128	256	128	256	64
Nodes HL 2	16	256	32	128	64
Nodes HL 3	64	64	256	64	16
Nodes HL 4	128	—	—	—	—
Loss	$2.99 \cdot 10^{-3}$	$2.24 \cdot 10^{-3}$	$4.07 \cdot 10^{-3}$	$5.03 \cdot 10^{-3}$	$3.90 \cdot 10^{-3}$

Surprisingly, 4 out of the 5 studies ended up recommending a best trial with 3 hidden layers, thus suggesting there is no need to add another one. The first hidden layer was once again the one with higher number of nodes across all studies. Regarding the remaining layers, their values have changed consistently along all the best trials, so their initially admitted values will stay the same.

Regarding the number of epochs, the models are tending to choose greater values for this hyperparameter, which makes sense taking into account the fact that the number of layers hasn't show inclination towards increasing, so better loss values are obtained by compensating with a higher number of epochs. Using 16 as the batch size presented overall better results, as the graph in Figure 5.20 confirms. At last, it can be inferred that the learning rates are showing higher values than for the Hertz MLP, so their range can also be narrowed.

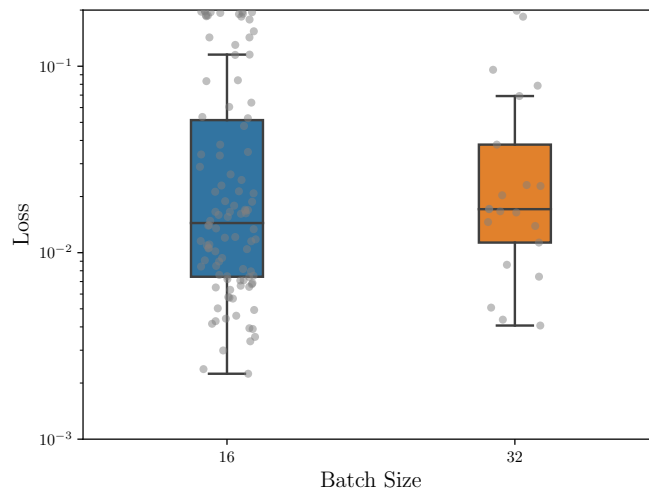


Figure 5.20: Loss distribution for trials with a batch size of 16 and 32. Trials with losses higher than 0.2 were eliminated for likely being associated with early pruning.

5.3.3.3 Analysis 3 - Defining Model's Width

Having settled the batch size and the final number of hidden layers in the second analysis, the following studies attempt to infer if there is any pattern for the model's width that consistently provides the best results, while allowing Optuna to also define the learning rate and number of epochs, as displayed in Table 5.26. The ranges of the learning rate, number of epochs and neurons in the first hidden layer were reduced based on the results from Analysis 2.

Table 5.26: Hyperparameters being optimized in the third analysis of the JKR MLP.

Hyperparameter	Type	Range
Learning Rate	Float	$[10^{-4}; 5 \cdot 10^{-3}]$
Num. of Epochs	Integer	[90, 100, ..., 130, 140]
Nodes in HL1	Categorical	[64, 128, 256]
Nodes in HL2	Categorical	[16, 32, 64, 128, 256]
Nodes in HL3	Categorical	[16, 32, 64, 128, 256]

5 studies with a reduced dataset of 10 000 instances were performed, with their results being summarised in Table 5.27.

Table 5.27: Best trials in each of the 5 studies performed in Analysis 3.

Hyperparameter	Study				
	1	2	3	4	5
Learning Rate	$6.38 \cdot 10^{-4}$	$1.31 \cdot 10^{-3}$	$2.45 \cdot 10^{-4}$	$2.71 \cdot 10^{-4}$	$7.24 \cdot 10^{-4}$
Num. of Epochs	110	120	140	120	90
Nodes HL 1	128	128	128	128	64
Nodes HL 2	32	128	16	128	16
Nodes HL 3	128	256	256	64	16
Loss	$2.98 \cdot 10^{-3}$	$8.05 \cdot 10^{-3}$	$4.49 \cdot 10^{-3}$	$2.93 \cdot 10^{-3}$	$4.16 \cdot 10^{-3}$

By interpreting the results, we can infer that the only point that the majority of the best trials have in common is the first hidden layer having 128 nodes. The remaining hyperparameters don't present noticeable similarities from one study to another, so from this analysis we'll select only the two trials (from studies 1 and 4) that provided lower losses, and try to decrease this value by optimizing their learning rates in the following analysis.

5.3.3.4 Analysis 4 - Optimizing the Learning Rates

This analysis will have the sole purpose of trying to improve the learning rates for the best trials of studies 1 and 4 of the last analysis. For simplification purposes, we'll now rename them as Models 4a) and 4b), respectively. Their architecture differs only in the number of nodes on the

second and third hidden layers, besides training for a different number of epochs. Being the only variable in study, the learning rate allowed range was kept the same as in the previous analysis.

For each model, 3 studies were conducted with the 10 000 instances dataset, from which the results depicted in Tables 5.28 and 5.29 were obtained.

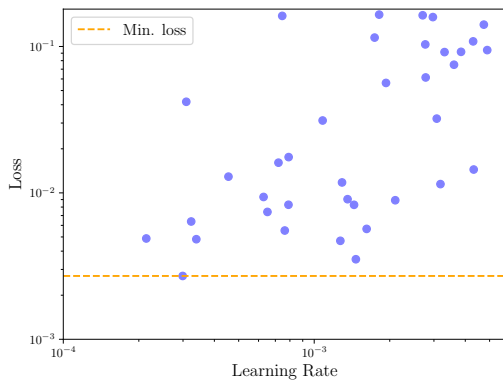
Table 5.28: Best trials for the 3 studies regarding Model 4a).

Hyperparameter	Study		
	1	2	3
Learning Rate	$3.390 \cdot 10^{-4}$	$1.466 \cdot 10^{-3}$	$2.990 \cdot 10^{-4}$
Loss	$4.83 \cdot 10^{-3}$	$3.52 \cdot 10^{-3}$	$2.70 \cdot 10^{-3}$

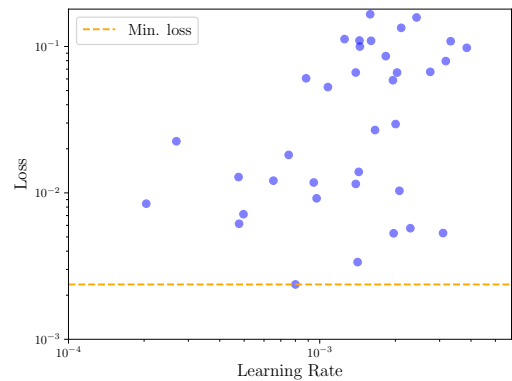
Table 5.29: Best trials for the 3 studies regarding Model 4b).

Hyperparameter	Study		
	1	2	3
Learning Rate	$1.415 \cdot 10^{-3}$	$1.968 \cdot 10^{-3}$	$8.003 \cdot 10^{-4}$
Loss	$3.37 \cdot 10^{-3}$	$5.30 \cdot 10^{-3}$	$2.37 \cdot 10^{-3}$

Comparing the two models, there were noticeable differences in the learning rates that generated the best results. Figure 5.21 displays the relation between the learning rate in each trial and the corresponding loss, for each model studied. While in Model 4a) we can see a slight relation between the increase in the learning rate resulting in a higher loss, in Model 4b) such correspondence can't be established. The low number of points presented in both graphs, when compared to the total number of trials, shows there was very intense pruning across all studies.



(a) Model 4a).



(b) Model 4b).

Figure 5.21: Loss as a function of the learning rate in Analysis 4.

5.3.4 Evaluating the Final Models

Having performed a thorough hyperparameter optimization, it is time to check if this process was effective. Since we currently don't possess any experimental withdraw curves with appropriate fitting to test our models on, this is the final model evaluation for the JKR MLPs.

In the Hertz MLP development, this similar evaluation was just an intermediate step, with the choice of the final models being only dependant on their experimental results. Hence, there were considered models that had been trained on a different number of instances for this comparison stage, although the main optimization refinements had been decided based on the common studies with a reduced dataset of 10 000 instances.

As the evaluation circumstances are different for the JKR MLP, we'll now only take into account models that were optimized using the reduced dataset, which means that the models from the preliminary studies in Analysis 1 won't be considered, so there is no bias in the comparisons due to the data used for hyperparameter optimization.

In addition to the base model, we'll also compare the best model from Analysis 2 (Model 2) and the two models that were selected from Analysis 3, whose learning rates were then improved in Analysis 4: Models 4a) and 4b). Table 5.30 recalls the hyperparameters of these models.

Table 5.30: Hyperparameters of the final JKR MLPs, highlighting the model with the best performance - Model 4a). The optimizer (Adam), loss (MSE) and activation (ReLU) were common to all the models.

Model	Nodes HL 1	Nodes HL 2	Nodes HL 3	Epochs	LR	Batch
Base Model	256	64	32	50	$5 \cdot 10^{-4}$	32
Model 2	256	256	64	130	$1.07 \cdot 10^{-3}$	16
Model 4a)	128	32	128	110	$2.99 \cdot 10^{-4}$	16
Model 4b)	128	128	64	120	$8.00 \cdot 10^{-4}$	16

To test all the models, 4 test folds of unseen data (6 000 instances each) were used in addition to the initial test set, similarly to what was done when choosing the right activation and loss functions. The average performance of the JKR MLPs on these folds, for both MAPE and percentage of curves with error lower than certain thresholds, is presented in Table 5.31 for E and in Table 5.32 for γ , where the results for the best model are highlighted.

Table 5.31: Average performance of the JKR MLP models on the 5 test folds, regarding material parameter E .

Model	$\epsilon_E < 10\%$	$\epsilon_E < 5\%$	$\epsilon_E < 2.5\%$	$\epsilon_E < 1\%$	$\bar{\epsilon}_E$	$\Delta\bar{\epsilon}_E$
Base Model	99.49%	97.64%	91.01%	54.18%	1.26%	-
Model 2	99.63%	98.44%	93.06%	58.28%	1.16%	-0.1%
Model 4a)	99.77%	99.31%	98.03%	85.70%	0.66%	-0.59%
Model 4b)	99.78%	98.11%	90.47%	48.63%	1.31%	+0.05%

Table 5.32: Average performance of the JKR MLP models on the 5 test folds, regarding material parameter γ .

Model	$\varepsilon_\gamma < 10\%$	$\varepsilon_\gamma < 5\%$	$\varepsilon_\gamma < 2.5\%$	$\varepsilon_\gamma < 1\%$	$\bar{\varepsilon}_\gamma$	$\Delta\bar{\varepsilon}_\gamma$
Base Model	99.62%	96.60%	73.74%	31.30%	1.89%	-
Model 2	99.86%	97.55%	83.71%	45.29%	1.47%	-0.42%
Model 4a)	99.82%	98.74%	88.71%	50.68%	1.27%	-0.62%
Model 4b)	99.35%	67.78%	13.37%	2.88%	4.37%	+2.48%

The hypertuning process proved to be successful, considering that for the optimization steps performed, all of them introduced improvements. However, one of the models in the fourth analysis - Model 4b) - had a poor performance on the energy of adhesion predictions, showing that its architecture wasn't appropriate for this problem, so it can be discarded. All the other models show very high accuracy on both parameters (slightly higher for the elastic modulus) and their behaviour was consistent throughout the 5 folds, as depicted in Figure 5.22.

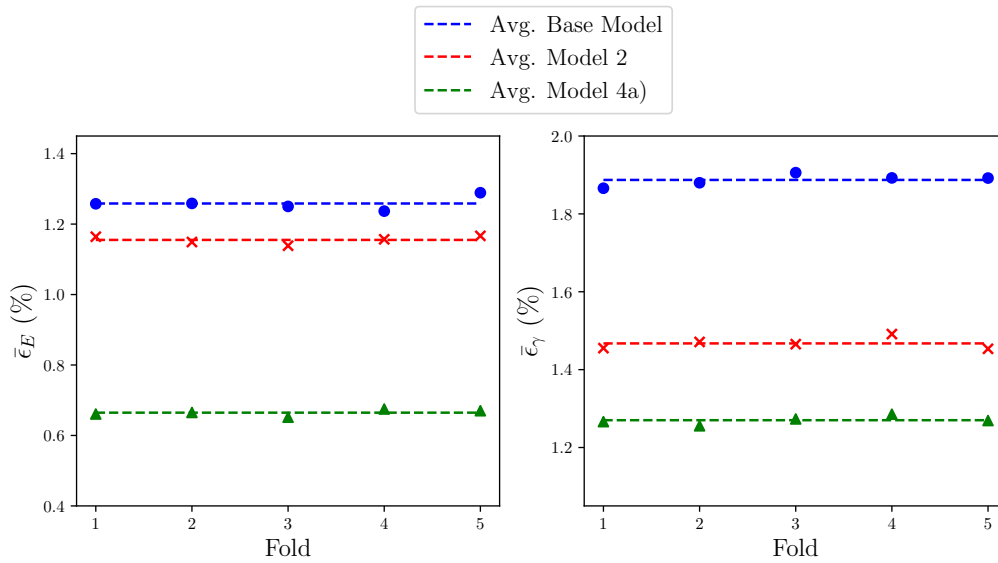


Figure 5.22: Relative error obtained for the 5 test folds, with the three best performing models.

Based only on the tests that were carried out with synthetic data, Model 4a) had the best performance overall, so it can be selected as our final model. Regarding the Young's modulus, besides the great average accuracy of 99.34%, it was able to predict more than 85% of the curves with an error lower than 1% for this parameter. However, the corresponding numbers for γ , despite illustrating a good model behaviour, don't reach the same magnitude, specially for the number of curves with an error lower than 1%. Hence, it is important to understand what are the real impacts of a certain relative error for E and for γ in the retraction curves, which will be done in the following section.

5.3.5 JKR MLP Concluding Remarks

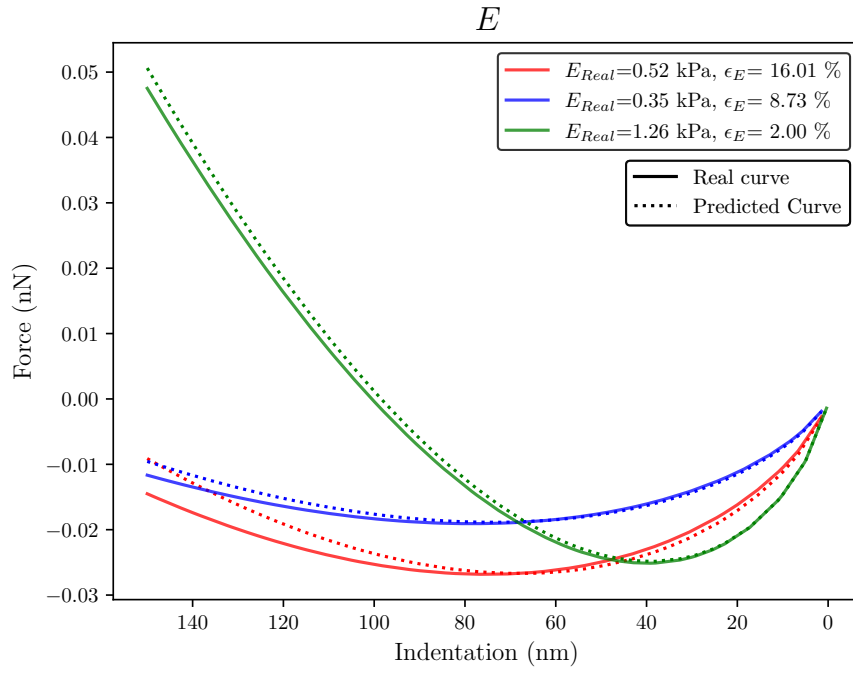
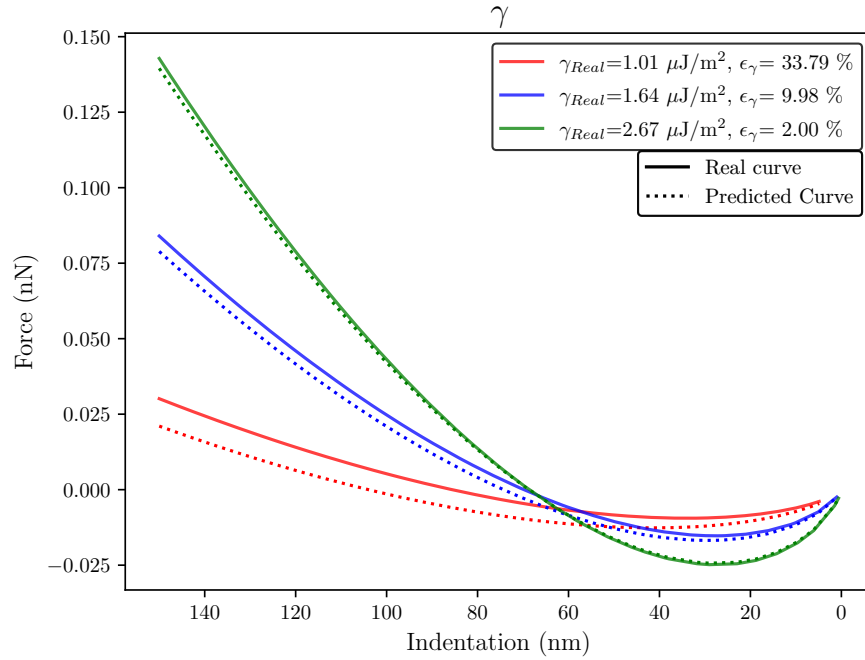
A thorough ML model development was once again applied, now for the prediction of surface properties based on AFM retraction curves. Stratifying the data revealed better results than when not opting for this approach, while a split with 70% of the initial dataset being used for training proved to be the better option.

Afterwards, six different configurations of loss and activation functions were tested. Here, MAPE was introduced as a potential loss function, since the target values were not on the same exact scale. However, using the Mean Squared Error still provided the best results. Through the course of this work, some experiments were also done regarding target normalization and standardization, but they didn't present improvements from simply using MSE with the raw targets. This can be due to the fact that the scales of both targets, despite not being the same, are not that different either. In cases where this difference is more significant, the same verification should be done, assessing what should be the best target normalization technique, whether using MAPE (or another loss function that produces normalized values), normalizing or standardizing the data. Another possibility would be defining different weights for the loss values and their gradients in the backward step, as PyTorch allows to perform backpropagation based on tensors, instead of always requiring a scalar value.

After optimizing all the hyperparameters that define our model, a 3 hidden layer network was reached. The studies performed in Optuna didn't show great evidence supporting that this number of layers should be increased, however, the good accuracy obtained was strongly linked to an enhancement in the number of epochs.

Not having any experimental data to verify the behaviour of these models in a real application is certainly a downsize, that leaves space to question if they would be able to generalize to experimental curves. Nonetheless, it has been shown that a Deep Learning approach is able to easily capture the shape of a Force-Indentation curve generated through JKR theory. By ensuring that these synthetic curves are good representations of nanoindentation retraction curves, it all indicates that this is a promising framework to study the adhesion energy of a wide range of samples, whose research is much more scarce than of the elastic modulus.

At last, there is a need to translate the relative error for both variables into the difference between actual and predicted Force-Indentation curves, to infer if one of the material parameters has a higher influence than the other. Figures 5.23 and 5.24 were obtained for different models developed along this chapter and show that a bad prediction of the elastic modulus will have a higher relevance in the accuracy of the predicted curve, so it is actually beneficial to have a higher accuracy for the Young's modulus, as verified in best performing models, including the final one, Model 4a).

Figure 5.23: Predicted and actual retraction curves for different relative errors of E .Figure 5.24: Predicted and actual retraction curves for different relative errors of γ .

Chapter 6

Testing on Experimental Data and Surface Mapping

In this chapter, the Hertz MLPs developed in chapter 5 will be tested on an experimental dataset obtained from AFM nanoindentations. At first, a description of the dataset is going to be made, followed by detailing the steps required for preprocessing the data, so it can be used as an input for the ANN models. Then, all the final models chosen for the Hertz MLP will be evaluated on this dataset to infer which one presents a better performance.

At last, a fractal surface generator will be used to create maps of the Young's modulus for synthetic surfaces, using the experimental stiffness values and F-I curves. The curves associated with each indentation point on a surface will serve as inputs for the MLPs. These MLPs will then predict the stiffness maps, showcasing the enhanced utility of the developed framework.

6.1 Introducing the Dataset

The experimental data available has information on 45 410 curves from AFM nanoindentations on biological soft tissues. For each sample, a map of indentation points was defined in its surface and a F-I curve was generated for each point. The dataset not only has the final F-I curves, but also the raw outputs, sensitivity, spring constant and cantilever deflection, amongst all the required variables to deal with the raw data. Key parameters such as the location of contact and detachment points are also available.

Despite having both approach and retraction parts of each curve, only the first ones were fitted with an appropriate contact model (Hertz), so only these will be used, as we have the corresponding Young's modulus for each approach curve. Of course some of the curves were affected by AFM artifacts and do not possess a regular shape or a shape similar to what would be expected from them, but they'll still be considered as inputs to our MLPs. The maximum indentation value is not the same for all curves, so this must be dealt with when preprocessing the data. There is a wide stiffness range along the different samples and Figure 6.1 displays some examples of experimental

approach curves for different stiffness values, while Figure 6.2 presents the Young's modulus distribution for all the instances.

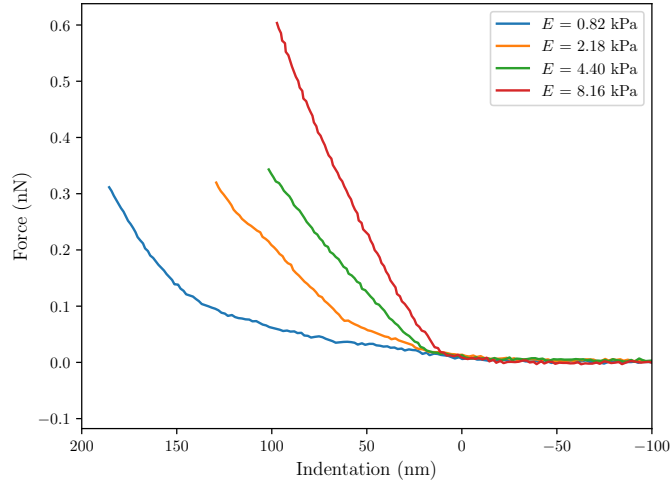


Figure 6.1: Examples of experimental approach curves in the available AFM nanoindentations data.

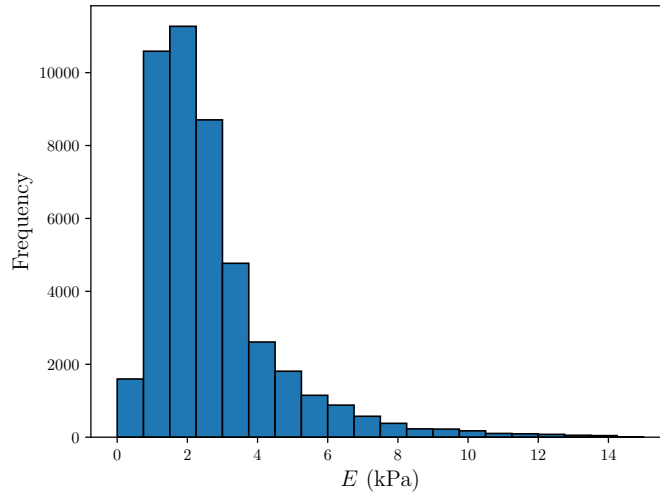


Figure 6.2: Young's modulus distribution over the experimental curves.

In addition to the distribution observed in Figure 6.2, there are some outliers that were not represented, for instance, negative stiffness values and values close to or even greater than 20 kPa. We can see that the large majority of the sampled points has a stiffness lower than 3 kPa. 89.9% of the data is clustered in the range of 0-5 kPa, around 8.9% between 5-10 kPa and only 1.2% of the curves have a stiffness higher than 10 kPa, so these will also be taken as outliers.

Now that the main features of the experimental dataset have been explored, it becomes crucial to preprocess the data and ensure its suitability as inputs for our ML models, as it will be done in the following section.

6.2 Preprocessing Experimental Data

As previously stated, every approach curve was fitted with Hertzian theory to predict their Young's modulus. However, this process was not accurate for every single curve, since some presented an unexpected behaviour. Each fitting was evaluated using the r^2 metric, calculated by:

$$r^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \quad (6.1)$$

where \hat{y}_i represents the predictions and \bar{y} the mean of the actual values. Approach curves that were fitted with an r^2 below 0.9 will be withdrawn from this analysis due to their likelihood of exhibiting experimental anomalies, as illustrated in Figure 6.3, where the presented curves show unexpected fluctuations for some indentation depths, except for the blue one, that despite having a regular shape, does not behave as predicted by Hertzian theory.

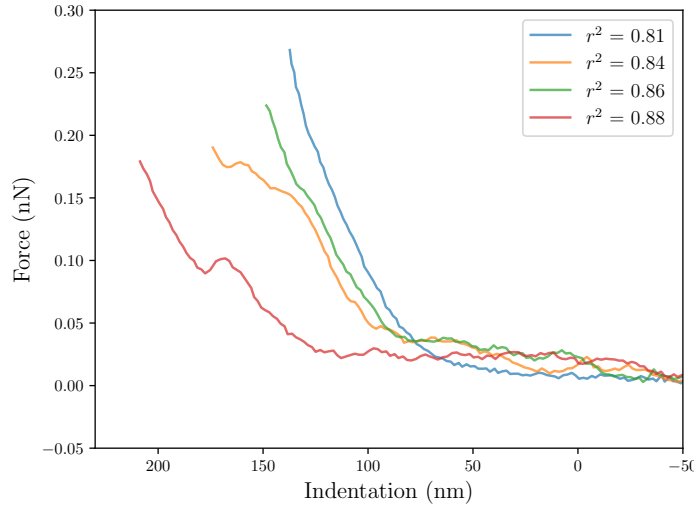


Figure 6.3: Examples of experimental approach curves that were fitted with an r^2 below 0.9.

The next elimination process involved removing curves with outlier stiffness, specifically those exceeding 10 kPa or displaying negative values for this parameter. Then, all the curves with a maximum indentation lower than 150 nm (the range defined for our MLP synthetic dataset) were dropped. This was the procedure that eliminated a greater number of curves (around 15 000), but it allows to analyse a larger portion of the remaining curves, extracting more information from them. In these remaining curves, the maximum indentation was set to 150 nm and the points with higher indentation were excluded.

Then, all points that had a negative indentation or force for the contact region were removed, since they would not be consistent with the Hertzian theory-based synthetic data in the same region. At last, force-indentation points were uniformly removed from each curve, until they reached the same 50 points as in the synthetic data, and the curves with less than that number of points were ignored. To conclude, the main statistics from the preprocessed experimental data are presented in Table 6.1.

Table 6.1: Number of curves and Young's modulus' statistics in the final experimental dataset.

Num. of curves	E (kPa)			
	Mean	Standard deviation	Minimum	Maximum
24304	1.84	0.77	0.19	5.93

A comparison between the experimental and synthetic approach curves is presented in Figure 6.4. If considering all the synthetic curves, since they have values that go up to 10 kPa, unlike the final experimental curves, their mean force for each indentation value will be higher (Figure 6.4a). However, if only the synthetic curves with E lower than 4 kPa are plotted (over 22 000 curves), it shows that there is enough data to comply with the selected experimental curves, although they have lesser values close to the minimum Young's modulus. Furthermore, it is worth noting that the synthetic dataset comprises 31 811 curves with an elastic modulus that falls within the range defined by the maximum and minimum values observed in the final experimental dataset.

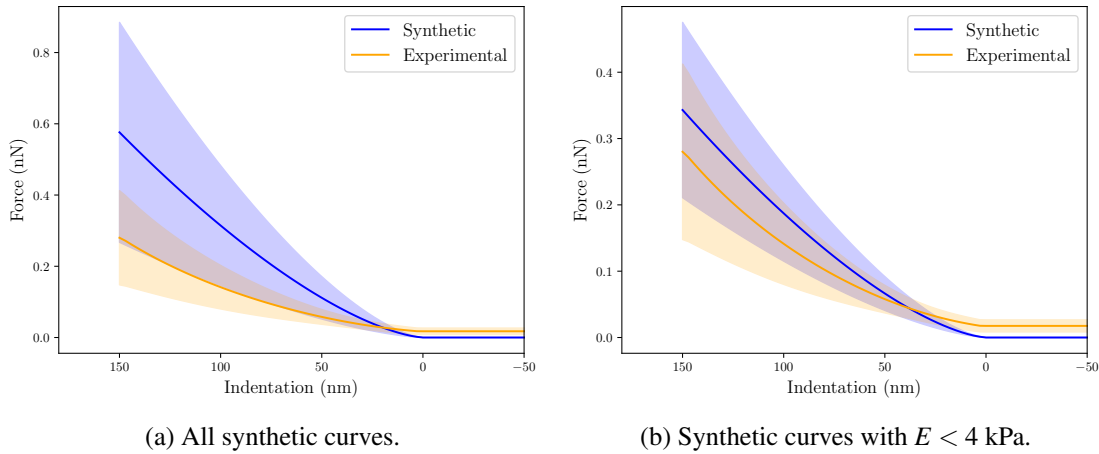


Figure 6.4: Mean synthetic and experimental curves, with errorbar representing the standard deviation.

6.3 Hertz MLPs Performance on Experimental Data

After having the AFM nanoindentation approach curves preprocessed and ready to be submitted into the developed Hertz MLPs, a new testing process can begin. At first, a similar approach from the one presented in Section 5.2.4 will be considered, where the base model and one model for each Optuna analysis will be tested and compared.

Then, the best performing model will be set as the final choice and it will be inferred if its performance would considerably increase if choosing only experimental curves with an r^2 higher than 0.9 (specifically 0.95 and 0.99). The models' performance on experimental data is detailed in Table 6.2.

Table 6.2: Performance of the models on the experimental test set. Note that the loss value in the Base Model was computed with the MSE, while Huber Loss was applied for the remaining. The best performing model is highlighted.

Model	Loss	$\varepsilon < 10\%$	$\varepsilon < 5\%$	$\varepsilon < 2.5\%$	$\varepsilon < 1\%$	$\bar{\varepsilon}$	$\Delta\bar{\varepsilon}$
Base Model	$31.83 \cdot 10^{-3}$	70.77%	34.41%	13.42%	4.51%	8.33%	-
Model 1	$10.88 \cdot 10^{-3}$	79.56%	47.01%	21.2%	8.3%	6.63%	-1.70%
Model 2	$12.12 \cdot 10^{-3}$	76.17%	39.33%	15.92%	5.63%	7.42%	-0.91%
Model 3	$7.46 \cdot 10^{-3}$	87.44%	65.61%	40.62%	17.70%	4.85%	-3.48%
Model 4	$6.66 \cdot 10^{-3}$	88.08%	67.33%	41.45%	18.50%	4.70%	-3.63%

There is now evidence that the hyperparameter optimization in Optuna produced good results, with the most optimized models having better generalization properties, which allowed the last model to have an accuracy $(1 - \bar{\varepsilon})$ of 95.3% on the experimental data.

In general, all models produced an MAPE lower than 10%, meaning that even the configuration defined for the base model allowed to build a robust MLP. With the successive optimization iterations, we were able to reduce the MAPE in 3.6%, but the most significant differences can be observed for the percentage of curves with a relative error below a certain threshold.

Comparing the model that achieved the best results with the initial one, 17% more curves had an ε below 10%, while this increase was in the order of the 33% for the curves with ε lower than 5% and 28% for curves with ε lower than 2.5%, which translate substantial improvements through our model development stage. Models 3 and 4 performed better than the rest, since they have more parameters: more hidden layers than the base model and Model 1, and more neurons in each layer than Model 2. This robustness gives the model better generalization capabilities. Considering all this, Model 4 will be set as our final proposed MLP and the only one being subjected to further analysis.

Moving on, the impact of the criterion for selecting good experimental curves will be studied, based on the r^2 error with which they were fitted. Figure 6.5 shows the relative error distributions if the final model was tested on the curves with $r^2 > 0.9$, $r^2 > 0.95$ and $r^2 > 0.99$. As expected, it can be seen that the model's accuracy increases with each narrowing of the selection criterion.

When dropping the approximately 1800 curves (still keeping 22 512 experimental curves) that have $0.9 < r^2 < 0.95$, there is an increase in 2.7% of the total curves with ε lower than 2.5%, which is not a very significant increase. The overall MAPE decreases from 4.7% to 4.0%.

On the other hand, if we keep narrowing down our selection, considering the 16 263 curves that were fitted with an r^2 higher than 0.99, the MLP performance experiences a more notable improvement. Focusing exclusively on these "top" curves, the model predicts more than half of them with a relative error below 2.5%, increasing the overall accuracy achieved from 95.3% to 96.7%.

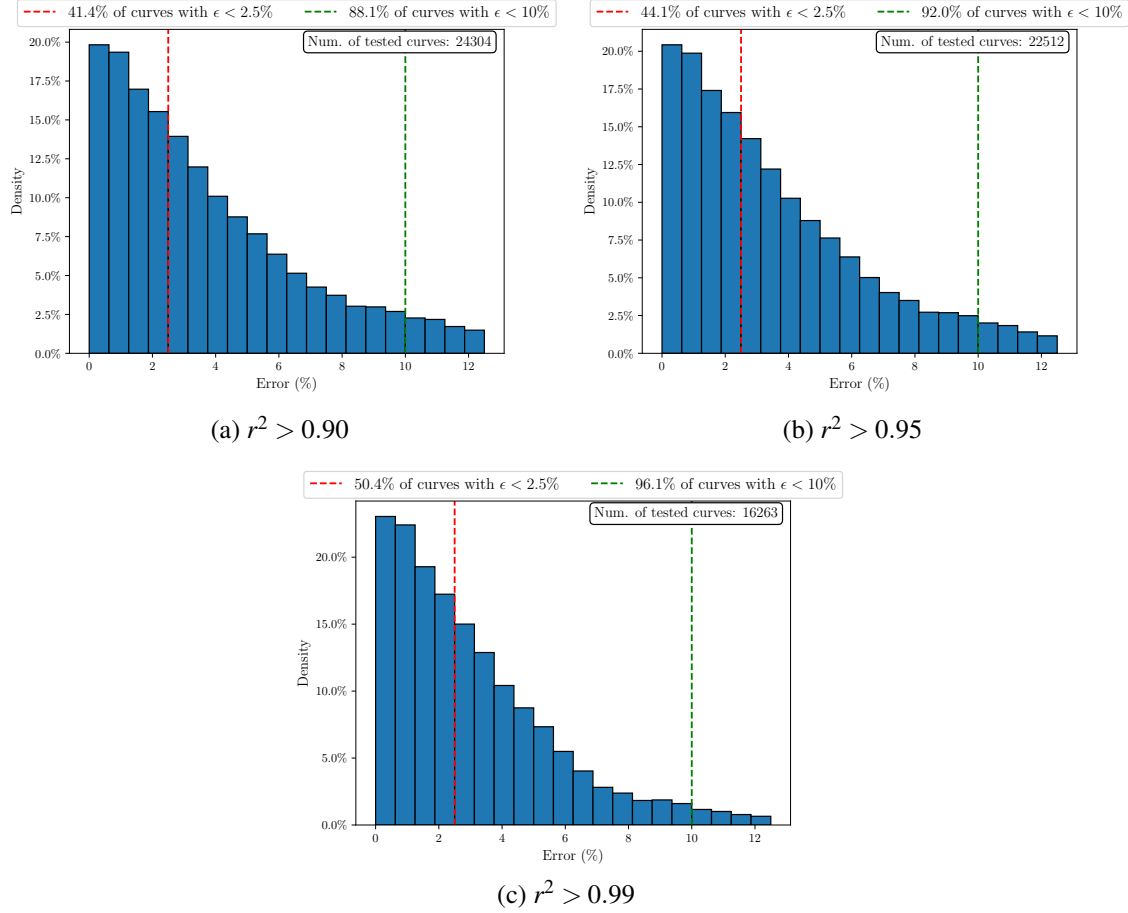


Figure 6.5: Comparison between the relative error distribution, when selecting experimental curves with different minimum r^2 fitting values.

Another crucial aspect to infer is if the curves that are being predicted with a higher error are concentrated in a specific range, specially since the Young's modulus distribution of the synthetic set doesn't correspond exactly to the one in the experimental data. It is also important to check if these curves with poor predictions have irregular shapes and were affected by AFM artifacts and the comparison of some experimental curves with the ones resulting from the predicted elastic modulus (and computed through the method detailed in section 5.1). Hence, Figure 6.6 shows the curves with the worst predictions and Figure 6.7 displays the difference between predicted and experimental curves for 3 examples.

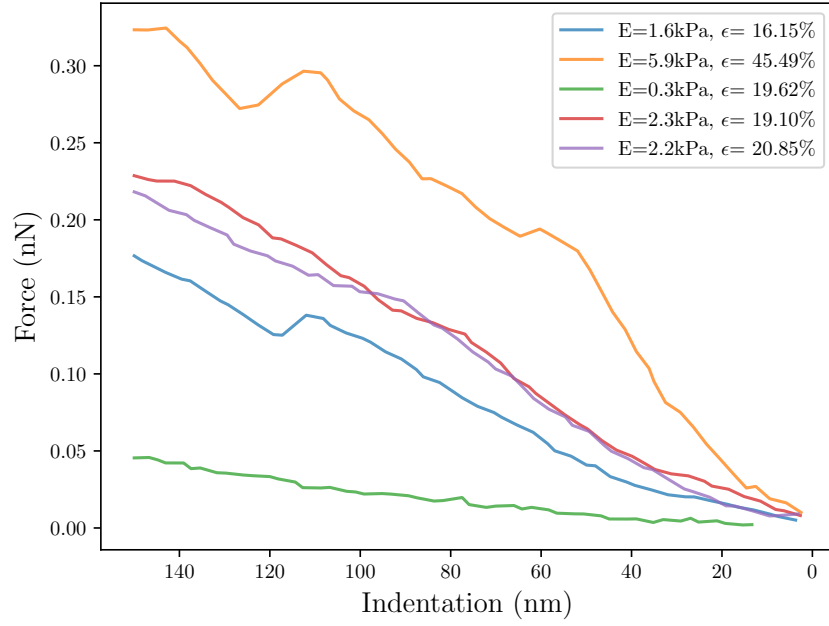


Figure 6.6: Experimental curves with $r^2 > 0.99$ with a high relative error.

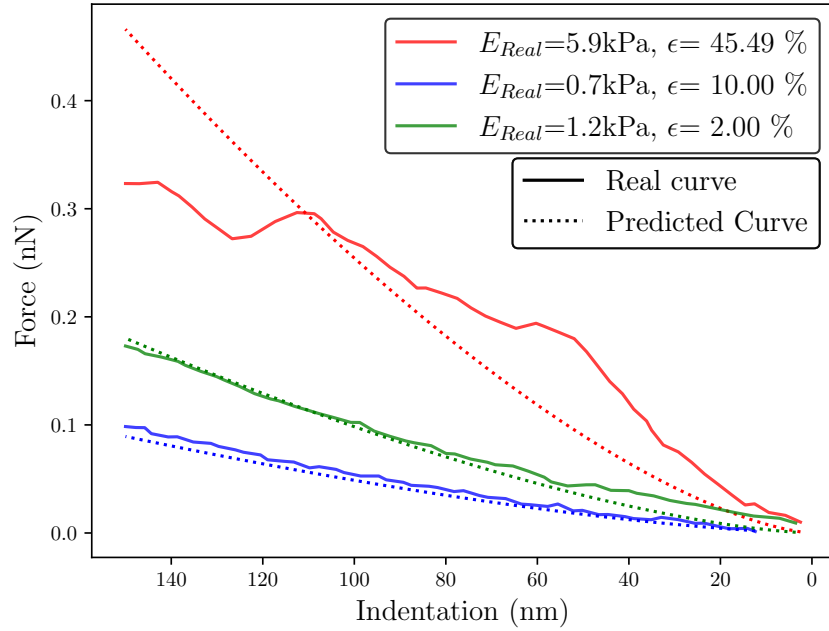


Figure 6.7: Comparison between predicted and experimental curves.

It can be concluded that the curves corresponding to the worst Young's modulus predictions are not focused in a single range of values, but it varies between 0.3 and 5.9 kPa, with solely one curve presenting an error significantly higher than 20%, which has a very irregular shape. This curve is also presented in Figure 6.7, where it is markedly noted that its behaviour is quite different from the predicted from Hertzian theory. The remaining curves shown in the same graph have high similarity between experimental and predicted ones.

6.4 Generating Stiffness Maps

In chapter 2, it was mentioned that one of the most common applications of AFM was to create maps for the sampled surfaces, regarding different parameters, from their height to a wide range of mechanical properties. Current AFM software allows not only to obtain mechanical properties from the selected F-I curves, by contact model fitting, but also to map them into the sample's surface. Hence, to enhance the utility of the current framework and capture a broader range of features from AFM software, stiffness predictions from individual curves will also be rearranged to stiffness maps. Since we currently don't possess any experimental data from topographic AFM that would allow to create surface height maps and together with F-I curves for each indented point, map the surface stiffness in different locations, these surfaces will be produced using the open-source *MATLAB* (MathWorks Inc., Natick, MA, USA) code proposed in (39). The code allows to generate surfaces with random roughness based on different height probability distributions and power spectrum.

In this section, a short description of the different parameters that can be tuned in the surface generator will be presented. Then it will be detailed the adopted procedure to transform 3D surfaces with only information on its dimensions, into stiffness maps, to be later predicted by our Hertz MLP.

6.4.1 Creating 3D Surfaces

To create authentic surface topographies, it is established that both the height and spatial distribution need to be defined. The height can be characterized using the height probability distribution (HPD), while the spatial distribution can be described by the power spectrum (PS). The code adopted in this framework has the advantage of conceiving surfaces with various HPDs, unlike many other generators that only adopt a Gaussian distribution. This is beneficial mainly due to the fact that different topographies will result in distinct behaviour when it comes to stiffness, adhesion or plastic deformation. In addition, it allows defining the HPD and PS separately, which is not common in available literature (37).

Three HPD functions are available: Gaussian, Bi-Gaussian and Weibull. The first is commonly used to represent smoother surfaces, the second often arises from multi-process manufacturing and the latter is suitable to model the result of wear.

Regarding the PS, we can choose from self-affine or exponential functions. Self-affine surfaces are characterized by displaying an unfolding symmetry, meaning that they are similar at different magnifications, i.e., in self-affine rough profiles, with the increase of magnification level,

new roughness profiles are added to the previous one, hence having scale-invariant properties. Nevertheless, the exponential autocorrelation function is the most common to create surface topographies.

Besides the specific coefficients for each HPD or PS, the anisotropy, low and high frequency cut-off ratios and the resolution of the map can also be modified when creating the surfaces.

6.4.2 From Height Distribution to Stiffness Maps

From the surfaces generated, only the height distribution over the x and y axes is known, so some correlation must be established to assign Young's modulus values for each point.

The method used in (8) proposes a linear relation between the applied load in each surface point and the normal stiffness, while in (38) the Hurst exponent and other roughness-related parameters affect this relation. Thus, there is no explicit method to compute the Young's modulus with the available information on the generated surfaces.

In view of all this, the surface height will be linearly correlated to the Young's modulus. For each surface, a maximum and minimum stiffness will be set corresponding to the highest and lowest points, respectively. For all the other points, their stiffness will be interpolated based on the boundary values defined. Then, to each point will be assigned the experimental F-I curve whose Young's modulus is the closest to the interpolated value for that same point. The only applied restriction is that the same curve can't be used twice. Choosing this way to distribute the elastic modulus values over the surface allows to visually assess if the model is performing worst for a specific interval. In addition, by linearly relating height with the elastic modulus it is ensured that there are no huge variations in this material parameter between two consecutive surface points, which is more representative of real surfaces.

All surfaces were generated with a unit anisotropy ratio and a resolution of 256×256 meaning that there were 256^2 sampled points for each surface. Before assigning stiffness values to the surface, it was downsized to a resolution of 32×32 , by picking points uniformly from its surface. With the established framework, a comparison between the actual and predicted E maps can be presented, together with mapping the error for each prediction over the surface.

6.5 Predicting Stiffness Maps with Hertz MLP

Throughout this section, it is going to be detailed all the pipeline that allows to generate maps with Young's modulus predictions, starting only with surface topography.

Along the course of this explanation, two Gaussian surfaces with self-affine properties will be used to illustrate all the described processes. The first depicts a smooth surface, that can be found, for instance, in biological soft tissues, while the second surface has a significantly higher roughness, that despite not being that frequently found in the kind of samples from which the experimental data were generated, it will serve as a demonstration of how this framework could be expanded to other types of materials. Nevertheless, this rough surface will still have an elastic modulus range within the same order of magnitude of the experimental data.

Figure 6.8 shows the created maps with their original resolution (256x256), that further on will be referred to as the "smooth" and "rough" surfaces.

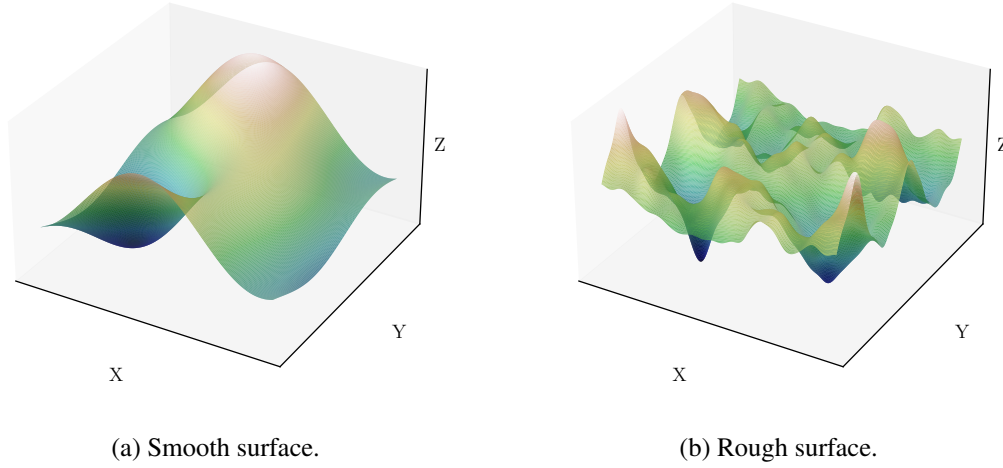


Figure 6.8: Initial synthetic surface maps.

To prevent from having to analyse all the sample's points from its surface, it is usual to reduce the dimensionality of the maps, so that the points in the condensed map not only represent themselves but also the region that was originally near them. Therefore, after uniformly selecting the appropriate points for dimensionality reduction, experimental F-I curves were attributed to them, as previously detailed. The smooth surface was considered to have stiffness values on the range of 0.2 kPa to 2.0 kPa, while higher values were chosen for the rough surface, ranging from 1.5 kPa to 5 kPa.

The F-I curves from each surface are used as inputs to our Hertz MLP, that predicts their corresponding Young's moduli and rearranges them back into a surface map format. Figures 6.9 and 6.10 represent a comparison between the maps of actual outputs and the predictions given by the model.

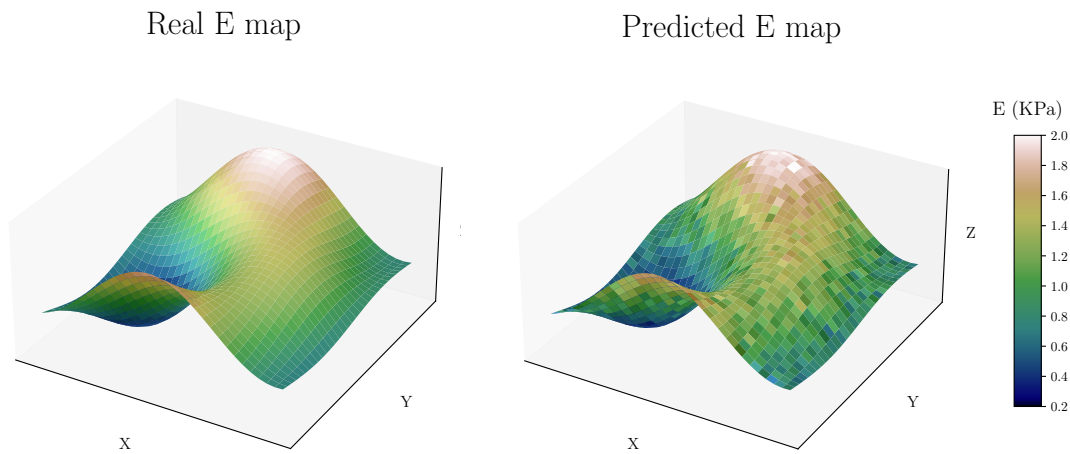


Figure 6.9: Real and predicted stiffness maps for the smooth surface.

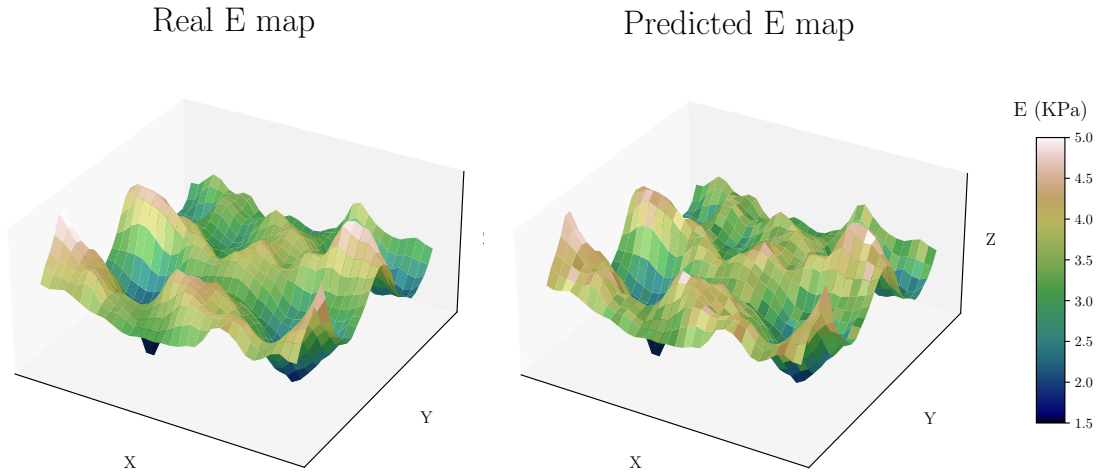


Figure 6.10: Real and predicted stiffness maps for the rough surface.

Overall, there is good agreement between the actual and the predicted maps, despite seeming that the model had more difficulty in predicting the lower values of the smooth surface. But to clarify which were the points where the model presented the greatest relative error, we can also map this variable, as done in Figure 6.11, while Figure 6.12 depicts the relative error distribution for each surface.

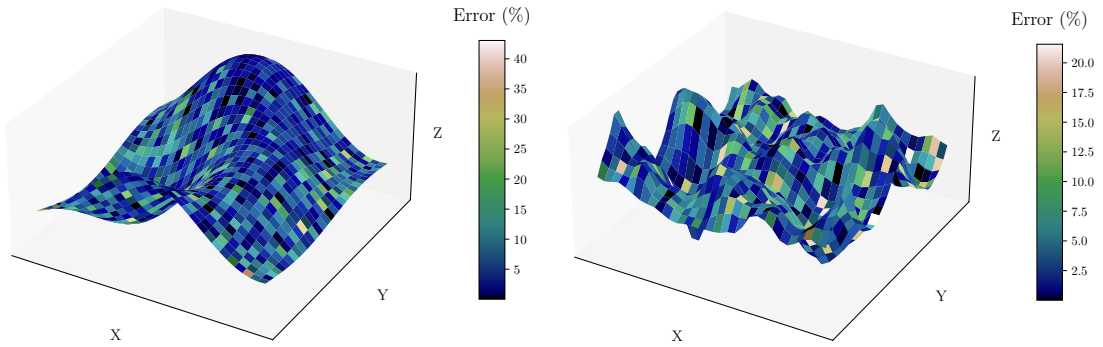


Figure 6.11: Relative error maps for the smooth and rough surfaces, respectively.

We can see that for both surfaces (hence, for different stiffness ranges) the relative error was lower than 10% for the most part. However, for the rough sample (higher E) no curve was predicted with an error much higher than 20%, while in the smooth sample (lower E) errors higher than 20% are observed more often, despite being a minority with little significance in the big picture. The graphs presented also confirm that the model has slight more difficulty in predicting

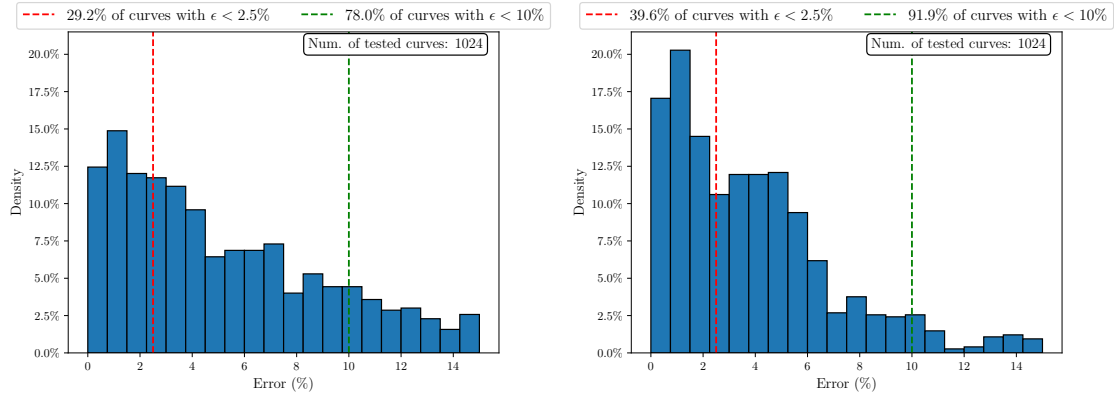


Figure 6.12: Relative error distribution for the predictions in the smooth (left) and rough (right) surfaces.

lower elastic moduli, regardless of reaching a relative error under 10% for almost 80% of the surface, which results in an average error of 6.5%. As for the higher modulus surface, the average error lowers to 4.2%, while over 90% of the curves are successfully predicted with an error lower than 10%.

To complement this analysis, Figure 6.13 identifies 3 points from the smooth surface, one that was very accurately predicted, another with an intermediate error and the point with the worst prediction.

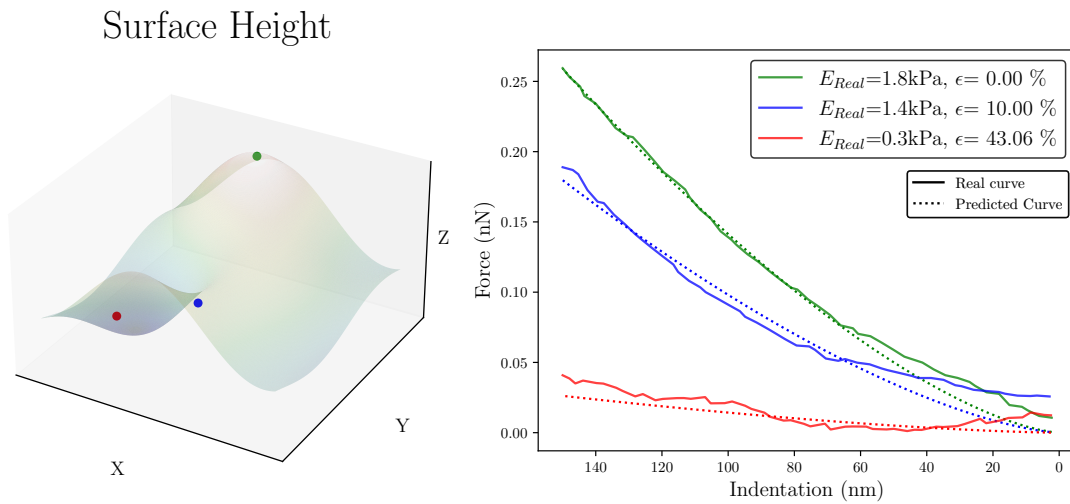


Figure 6.13: Actual and predicted curves for 3 points identified in the smooth surface.

A sample with 0.0% error was reached, for a value closer to the maximum Young's modulus in the smooth surface, but once again it is confirmed that the worst prediction is close to the minimum stiffness of the range in which the model was trained. Nonetheless, its predicted curve is still very close to the experimental one.

To sum up, this section has demonstrated an additional path for exploring the presented framework. In addition to its capability to predict surface properties from individual force-indentation curves, the framework's ability to map these properties represents a significant advancement in the comprehensive characterization of samples. This approach offers a computationally optimized and time-efficient solution for fully characterizing samples, enhancing the efficiency and effectiveness of the analysis process.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Throughout this work, two MLP models were designed from scratch, with the goal of predicting surface properties from AFM nanoindentation curves.

In the first stage, synthetic F-I curves were generated based on Hertz and JKR contact theories, with parameters that would allow them to replicate the curves that make up the available experimental dataset. After that, the models started being developed using a similar strategy. The advantages of stratifying the data in the split were proven and the choice of an intermediate split ratio (compared to the three tested values) was demonstrated to be the better option for both models. An initial brute-force approach was used to try and find the most appropriate loss and activation functions. Then, a finer optimization technique was applied by implementing Optuna, that allowed to sequentially select the hyperparameters that would translate into the best model performance.

Regarding the Hertz MLP, an intermediate comparison between the final model and the previous ones was carried out, by evaluating their performance on the synthetic test set. The final model didn't prove to have the best behaviour on the synthetic data, mainly because some of the models with sub-optimal parameters had been optimized using the entire initial dataset, while the best hyperparameters were achieved by optimizing the model with a reduced set. This indicated the initial parameters could be leading to overfitting to the synthetic dataset, which was to be later confirmed when testing the models on the experimental set.

As for the JKR MLP, in the absence of experimental data to evaluate it, its testing on synthetic data had to be more robust, so the final proposed models were evaluated on 5 unseen test folds. A model that accurately predicts the Young's modulus at a rate of 99.3% and adhesion energy at 98.7% was reached, showing that the Multilayer Perceptron developed can capture a F-I curve that complies with JKR theory. However, its validation on real data is still required, such as future adjustments on how to define the detachment point on retraction curves creation, to increase their affinity with experimental curves.

What was initially thought concerning the first framework, was confirmed when testing the

successive Hertz MLPs in the experimental dataset. The model that went through more hypertuning stages presented the best behaviour, accurately predicting the more than 24 000 curves fitted with a $r^2 > 0.9$ with an accuracy of 95.3%. When selecting only the top curves ($r^2 > 0.99$), this value increased to 96.7%, with more than half the curves being predicted with a relative error under 2.5%. These results validate the use of Deep Learning strategies to predict surface properties from experimental AFM nanoindentation curves, trained only on synthetic data, so they can be later rearranged into maps illustrating the distribution of these properties along the surface. However, the Hertz MLP showed a worse behaviour for curves with lower elastic moduli, thus needing more refinement in its hyperparameter optimization, possibly even adding more complexity to the NN, to see its results further improved.

7.2 Future Work

In future improvements of this framework, it would be important to try different approaches in data generation, possibly using other contact models, to infer if it would increase the model's ability to make predictions on experimental data. Still concerning synthetic data generation, it could be useful to test if reinforcing the data in ranges where the model had weaker performance would impact its results on the real curves. If possible, the model should be tested on more experimental datasets for other ranges of the studied surface properties.

Another addition that would allow to better assess the model's performance, would be to define a more accurate process to extract the bad curves from the dataset, identifying if any AFM artifacts had any negative impact on each curve, instead of relying solely on the r^2 fitting metric.

For the JKR MLP, it would be important to be tested on experimental data, to check its real accuracy or if many changes would be required for it to be applied in real applications.

It would also be of great interest to define a reliable method to define the detachment point in the retraction curves, which throughout this work was considered to be the same as the contact point in the approach curves, and it is not generally the case in real applications. In addition, developing a Deep Learning model capable of predicting the location of the contact point for each curve could be of great interest, since this is one of the current main difficulties in AFM. Furthermore, future improvements for this model could involve its generalization to a broader spectrum of tip radii, allowing it to be applied to a wide range of values instead of being limited to the radius specified during synthetic data generation.

References

- [1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Op-tuna: A next-generation hyperparameter optimization framework. 2019.
- [2] P. Belardinelli, A. Chandrashekar, R. Wiebe, F. Alijani, and S. Lenci. Machine learning to probe modal interaction in dynamic atomic force microscopy. *Mechanical Systems and Signal Processing*, 179:109312, 2022. ISSN 0888-3270. doi: <https://doi.org/10.1016/j.ymssp.2022.109312>. URL <https://www.sciencedirect.com/science/article/pii/S0888327022004502>.
- [3] G. Binnig, H. Rohrer, Ch. Gerber, and E. Weibel. Surface studies by scanning tunneling microscopy. *Phys. Rev. Lett.*, 49:57–61, Jul 1982. doi: 10.1103/PhysRevLett.49.57.
- [4] G. Binnig, C. F. Quate, and Ch. Gerber. Atomic force microscope. *Phys. Rev. Lett.*, 56: 930–933, Mar 1986. doi: 10.1103/PhysRevLett.56.930.
- [5] Marco Bontempi, Francesca Salamanna, Rosario Capozza, Andrea Visani, Milena Fini, and Alessandro Gambardella. Nanomechanical mapping of hard tissues by atomic force microscopy: An application to cortical bone. *Materials*, 15(21), 2022. ISSN 1996-1944. doi: 10.3390/ma15217512. URL <https://www.mdpi.com/1996-1944/15/21/7512>.
- [6] Fadi Boutros, Vitomir Struc, Julian Fierrez, and Naser Damer. Synthetic data for face recognition: Current state and future prospects. *Image and Vision Computing*, 135:104688, 2023. ISSN 0262-8856. doi: <https://doi.org/10.1016/j.imavis.2023.104688>. URL <https://www.sciencedirect.com/science/article/pii/S0262885623000628>.
- [7] B J Briscoe, K S Sebastian, and M J Adams. The effect of indenter geometry on the elastic response to indentation. *Journal of Physics D: Applied Physics*, 27(6):1156, jun 1994. doi: 10.1088/0022-3727/27/6/013. URL <https://dx.doi.org/10.1088/0022-3727/27/6/013>.
- [8] C Campaña, B N J Persson, and M H Müser. Transverse and normal interfacial stiffness of solids with randomly rough surfaces. *Journal of Physics: Condensed Matter*, 23(8): 085001, feb 2011. doi: 10.1088/0953-8984/23/8/085001. URL <https://dx.doi.org/10.1088/0953-8984/23/8/085001>.
- [9] Elisabetta Ada Cavalcanti-Adam, Tova Volberg, Alexandre Micoulet, Horst Kessler, Benjamin Geiger, and Joachim Pius Spatz. Cell spreading and focal adhesion dynamics are regulated by spacing of integrin ligands. *Biophysical Journal*, 92(8):2964–2974, 2007. ISSN 0006-3495. doi: <https://doi.org/10.1529/biophysj.106.089730>. URL <https://www.sciencedirect.com/science/article/pii/S0006349507710998>.

- [10] Rogerio Colaço and Patricia Carvalho. *Atomic Force Microscopy in Bioengineering Applications*, pages 397–430. Scanning Probe Microscopy in Nanoscience and Nanotechnology 3, 01 2013. ISBN 978-3-642-25413-0 (Print) 978-3-642-25414-7 (Online). doi: 10.1007/978-3-642-25414-7_15.
- [11] Liz Ivoneth Del Cid. *A Discrete Element Methodology for the Analysis of Cohesive Granular Bulk Solid Materials*. PhD thesis, Colorado School of Mines, 2015.
- [12] Peter Eaton and Paul West. *Atomic Force Microscopy*. Oxford University Press, Oxford, UK, 2010. ISBN 978-0-19-957045-4.
- [13] Yu.M. Efremov, D.V. Bagrov, M.P. Kirpichnikov, and K.V. Shaitan. Application of the johnson–kendall–roberts model in afm-based mechanical measurements on cells and gel. *Colloids and Surfaces B: Biointerfaces*, 134:131–139, 2015. ISSN 0927-7765. doi: <https://doi.org/10.1016/j.colsurfb.2015.06.044>. URL <https://www.sciencedirect.com/science/article/pii/S0927776515300047>.
- [14] JPS Ferreira, M Kuang, M Marques, MPL Parente, MS Damaser, and RM Natal Jorge. On the mechanical response of the actomyosin cortex during cell indentations. *Biomechanics and modeling in mechanobiology*, 19(6):2061–2079, 2020. doi: 10.1007/s10237-020-01324-5. URL <https://doi.org/10.1007/s10237-020-01324-5>.
- [15] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media, Sebastopol, CA, 2019. ISBN 978-1492032649.
- [16] Heinrich Hertz. Ueber die berührung fester elastischer körper. 1882.
- [17] K. L. Johnson, K. Kendall, and A. D. Roberts. Contact mechanics. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 324(1558):301–313, 1971.
- [18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015. doi: 10.48550/arXiv.1412.6980.
- [19] Viktor Kocur, Veronika Hegrová, Marek Patočka, Jan Neuman, and Adam Herout. Correction of afm data artifacts using a convolutional neural network trained with synthetically generated data. *Ultramicroscopy*, 246:113666, 2023. ISSN 0304-3991. doi: <https://doi.org/10.1016/j.ultramic.2022.113666>. URL <https://www.sciencedirect.com/science/article/pii/S0304399122001851>.
- [20] S V Kontomaris and A Malamou. A novel approximate method to calculate the force applied on an elastic half space by a rigid sphere. *European Journal of Physics*, 42(2):025010, feb 2021. doi: 10.1088/1361-6404/abccfb.
- [21] S.V. Kontomaris, A. Georgakopoulos, A. Malamou, and A. Stylianou. The average young’s modulus as a physical quantity for describing the depth-dependent mechanical properties of cells. *Mechanics of Materials*, 158:103846, 2021. ISSN 0167-6636. doi: <https://doi.org/10.1016/j.mechmat.2021.103846>. URL <https://www.sciencedirect.com/science/article/pii/S0167663621000995>.
- [22] S.V. Kontomaris, A. Malamou, and A. Stylianou. The hertzian theory in afm nanoindentation experiments regarding biological samples: Overcoming limitations in data processing. *Micron*, 155:103228, 2022. ISSN 0968-4328. doi: <https://doi.org/10.1016/j.micron>.

- 2022.103228. URL <https://www.sciencedirect.com/science/article/pii/S0968432822000245>.
- [23] Malgorzata Lekka. Discrimination between normal and cancerous cells using afm. *Bio-NanoSci*, 6:65–80, 2016. doi: 10.1007/s12668-016-0191-3.
- [24] Malgorzata Lekka, Piotr Laidler, Danuta Gil, Janusz Lekki, Zbigniew Stachura, and Andrzej Z Hryniewicz. Elasticity of normal and cancerous human bladder cells studied by scanning force microscopy. *European Biophysics Journal*, 28(4):312–316, 1999. doi: 10.1007/s002490050213.
- [25] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.
- [26] Joana Moreira, Manuel Simões, Luis Melo, and Filipe Mergulhão. Escherichia coli adhesion to surfaces—a thermodynamic assessment. *Colloid and Polymer Science*, 10 2014. doi: 10.1007/s00396-014-3390-x.
- [27] Philipp Müller, Shadi Abuhattum, Sarah Möllmert, et al. nanite: using machine learning to assess the quality of atomic force microscopy-enabled nano-indentation data. *BMC Bioinformatics*, 20(1):465, 2019. doi: 10.1186/s12859-019-3010-3.
- [28] Inc. Preferred Networks. Optuna: A hyperparameter optimization framework. <https://optuna.org/>. Accessed on 17/04/2023.
- [29] Izaak Neutelings. Neural Networks. https://tikz.net/neural_networks/, 2022. Accessed 2023-05-12.
- [30] Linh Thi Phuong Nguyen and Bernard Haochih Liu. Machine learning framework for determination of elastic modulus without contact model fitting. *International Journal of Solids and Structures*, 256:111976, 2022. ISSN 0020-7683. doi: <https://doi.org/10.1016/j.ijsolstr.2022.111976>. URL <https://www.sciencedirect.com/science/article/pii/S0020768322004292>.
- [31] Thao Nguyen and Yu Gu. Investigation of cell-substrate adhesion properties of living chondrocyte by measuring adhesive shear force and detachment using afm and inverse fea. *Scientific Reports*, 6:38059, 2016. doi: 10.1038/srep38059.
- [32] Michael Nielsen. *Neural Networks and Deep Learning*. davidsandberg, 2015. URL <http://neuralnetworksanddeeplearning.com>. Online book.
- [33] Maria Rossella Nobile, Liberata Guadagno, Carlo Naddeo, Luigi Vertuccio, and Marialuigia Raimondo. Graphene/epoxy resins: Rheological behavior and morphological analysis by atomic force microscopy (afm). *Materials Today: Proceedings*, 34:160–163, 2021. ISSN 2214-7853. doi: <https://doi.org/10.1016/j.matpr.2020.02.139>. URL <https://www.sciencedirect.com/science/article/pii/S2214785320308907>.
- [34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. <https://pytorch.org>, 2019.

- [35] Alvaro Paulo and Ramiro García-García. High-resolution imaging of antibodies by tapping-mode atomic force microscopy: Attractive and repulsive tip-sample interaction regimes. *Biophysical journal*, 78:1599–605, 04 2000. doi: 10.1016/S0006-3495(00)76712-9.
- [36] George M. Pharr and Alexey Bolshakov. Understanding nanoindentation unloading curves. *Journal of Materials Research*, 17(11):2660–2671, 2002. doi: 10.1557/JMR.2002.0386.
- [37] Francesc Pérez Ràfols and Andreas Almqvist. Generating randomly rough surfaces with given height probability distribution and power spectrum. *Tribology International*, 131:591–604, 2019. ISSN 0301-679X. doi: <https://doi.org/10.1016/j.triboint.2018.11.020>.
- [38] Francesc Pérez Ràfols and Andreas Almqvist. On the stiffness of surfaces with non-gaussian height distribution. *Scientific Reports*, 11, 01 2021. doi: 10.1038/s41598-021-81259-8.
- [39] Francesc Pérez Ràfols and Andreas Almqvist. Fractal roughness generator, 02 2023. URL <https://www.mathworks.com/matlabcentral/fileexchange/129469-fractal-surface-generator>. Accessed 2023-05-15.
- [40] Jean-Francois Rajotte, Robert Bergen, David L. Buckeridge, Khaled El Emam, Raymond Ng, and Elissa Strome. Synthetic data as an enabler for machine learning applications in medicine. *iScience*, 25(11):105331, 2022. ISSN 2589-0042. doi: <https://doi.org/10.1016/j.isci.2022.105331>. URL <https://www.sciencedirect.com/science/article/pii/S2589004222016030>.
- [41] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=ryQu7f-RZ>.
- [42] David C. Roache, Clifton H. Bumgardner, Yunya Zhang, David Edwards, David DeGonia, Bryan Rock, and Xiaodong Li. Chemo-mechanical characterization of phosphorus and sulfur containing ashless tribofilms on hypoid gear teeth. *Tribology International*, 158:106926, 2021. ISSN 0301-679X. doi: <https://doi.org/10.1016/j.triboint.2021.106926>. URL <https://www.sciencedirect.com/science/article/pii/S0301679X21000748>.
- [43] Herbert Robbins and Sutton Monro. Stochastic approximation and recursive algorithms and applications. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.
- [44] Suprakas Sinha Ray. *Techniques for characterizing the structure and properties of polymer nanocomposites*, pages 74–88. 12 2013. ISBN 9780857097774. doi: 10.1533/9780857097828.1.74.
- [45] Ian N. Sneddon. The relation between load and penetration in the axisymmetric boussinesq problem for a punch of arbitrary profile. *International Journal of Engineering Science*, 3(1):47–57, 1965. ISSN 0020-7225. doi: [https://doi.org/10.1016/0020-7225\(65\)90019-4](https://doi.org/10.1016/0020-7225(65)90019-4). URL <https://www.sciencedirect.com/science/article/pii/0020722565900194>.
- [46] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

- [47] Park Systems. Park AFM Probe Guide. https://www.parksystems.com/images/media/brochures/probe/Park_AFM_ProbeGuide_190620E32AB.pdf, 2019.
- [48] Norio Taniguchi. On the basic concept of nanotechnology. *Proceedings of the International Conference on Production Engineering*, pages 18–23, 1974.
- [49] Alan Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.
- [50] Gunjan Tyagi, Zain Ahmad, Luca Pellegrino, Luis M.G. Torquato, Eric S.J. Robles, and João T. Cabral. Effect of surface energy on the removal of supported triglyceride films by a flowing surfactant solution. *Surfaces and Interfaces*, page 102992, 2023. ISSN 2468-0230. doi: <https://doi.org/10.1016/j.surfin.2023.102992>. URL <https://www.sciencedirect.com/science/article/pii/S2468023023003620>.
- [51] Bert Voigtländer. *Atomic Force Microscopy*. Springer, Second edition, 2019. ISBN 978-3-030-13653-6.
- [52] Joshua R. Waite, Sin Yong Tan, Homagni Saha, Soumik Sarkar, and Anwesha Sarkar. Few-shot deep learning for afm force curve characterization of single-molecule interactions. *Patterns*, 4(1):100672, 2023. ISSN 2666-3899. doi: <https://doi.org/10.1016/j.patter.2022.100672>. URL <https://www.sciencedirect.com/science/article/pii/S2666389922003191>.
- [53] Papers with Code. Papers with code trends. <https://paperswithcode.com/trends>. Accessed 2023-03-02.
- [54] Gang Sheng Chen Xiandong Liu. *Friction Dynamics. Principles and Applications*. Woodhead Publishing in Mechanical Engineering. Woodhead Publishing, first edition, 2016. ISBN 9780081002858.
- [55] Mohd Zaki, S. Kasimuthumaniyan, Sourav Sahoo, Jayadeva, Nitya Nand Gosvami, and N. M. Anoop Krishnan. Interpretable machine learning approach for identifying the tip sharpness in atomic force microscopy. *Scripta Materialia*, 221:114965, 2022. ISSN 1359-6462. doi: <https://doi.org/10.1016/j.scriptamat.2022.114965>. URL <https://www.sciencedirect.com/science/article/pii/S1359646222004602>.
- [56] Yuyao Zhang, Xiaoying Zhu, and Baoliang Chen. Adhesion force evolution of protein on the surfaces with varied hydration extent: Quantitative determination via atomic force microscopy. *Journal of Colloid and Interface Science*, 608:255–264, 2022. ISSN 0021-9797. doi: <https://doi.org/10.1016/j.jcis.2021.09.131>. URL <https://www.sciencedirect.com/science/article/pii/S0021979721015903>.
- [57] Xinyao Zhu, Rui Qin, Kaige Qu, Zuobin Wang, Xuexia Zhao, and Wei Xu. Atomic force microscopy-based assessment of multimechanical cellular properties for classification of graded bladder cancer cells and cancer early diagnosis using machine learning analysis. *Acta Biomaterialia*, 158:358–373, 2023. ISSN 1742-7061. doi: <https://doi.org/10.1016/j.actbio.2022.12.035>. URL <https://www.sciencedirect.com/science/article/pii/S1742706122008388>.

Appendix A

Implementing Optuna in Pytorch

A.1 Defining the Model

```
# Set cuda as the preferred device
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'

# Define the model to be called during the studies
def define_model(trial):

    layers = []
    # Suggests the number of layers
    n_layers = trial.suggest_int("n_layers", 2, 4)
    # Define the input shape of the model
    in_features = input_shape[0] * input_shape[1]

    for i in range(n_layers):
        if i == 0:
            layers.append(nn.Flatten())
            # Suggests the number of neurons in the layer
            out_features = trial.suggest_categorical("n_units_l{}".format(i),
                                                    [16, 32, 64, 128, 256])
            layers.append(nn.Linear(in_features, out_features))
            layers.append(nn.LeakyReLU())
            in_features = out_features
        layers.append(nn.Linear(in_features, 1))
    return nn.Sequential(*layers)
```

A.2 Defining the Objective Function

Define the objective function

```
def objective(trial):
    torch.manual_seed(42)
    model_Hertz = define_model(trial).to(DEVICE)
    # Suggests values for the studied hyperparameters
    learning_rate = trial.suggest_float("learning_rate", 1e-5, 1e-2, log=True)
    n_epochs = trial.suggest_int('n_epochs', 10, 100, step=10)
    optimizer_name = trial.suggest_categorical("optimizer", ["Adam", "SGD"])
    batch_size = trial.suggest_categorical("batch_size", [16, 32, 64, 128])

    train_loader=DataLoader(train_data, batch_size=batch_size, shuffle=True)
    optimizer = getattr(optim, optimizer_name)(model_Hertz.parameters(),
                                                lr=learning_rate)

    loss_fn = nn.HuberLoss()
    for epoch in range(n_epochs):
        model_Hertz.train(True)
        for i, data in enumerate(train_loader):
            inputs, labels = data
            optimizer.zero_grad()
            outputs = model_Hertz(inputs.to(DEVICE))
            loss = loss_fn(outputs, labels.to(DEVICE))
            loss.backward()
            optimizer.step()

        # Evaluation
        model_Hertz.eval()
        running_vloss = 0.0
        with torch.no_grad():
            for i, vdata in enumerate(valid_loader):
                vinputs, vlabels = vdata
                voutputs = model_Hertz(vinputs.to(DEVICE))
                vloss = loss_fn(voutputs, vlabels.to(DEVICE))
                running_vloss += vloss

            loss = running_vloss / (i + 1)
        trial.report(loss.item(), epoch)
        # Handle pruning based on the intermediate value.
        if trial.should_prune():
            raise optuna.exceptions.TrialPruned()
    return loss.item()
```

```
if __name__ == "__main__":
    study_name = "hertz_Huber_LeakyReLU"
    study = optuna.create_study(direction="minimize",
                                pruner=optuna.pruners.MedianPruner(n_startup_trials=5,
                                                                    n_warmup_steps=2),
                                study_name=study_name)
    study.optimize(objective, n_trials=100)
    pruned_trials = study.get_trials(deepcopy=False,
                                     states=[TrialState.PRUNED])
    complete_trials = study.get_trials(deepcopy=False,
                                       states=[TrialState.COMPLETE])

    print("Study statistics: ")
    print("  Number of finished trials: ", len(study.trials))
    print("  Number of pruned trials: ", len(pruned_trials))
    print("  Number of complete trials: ", len(complete_trials))
    print("Best trial:")
    trial = study.best_trial
    print("  Value: ", trial.value)
    print("  Params: ")
    for key, value in trial.params.items():
        print("    {}: {}".format(key, value))
    # Retrieve the losses from each trial
    losses = [trial.value for trial in study.trials]
```
