Master Geomechanics, Civil Engineering and Risks

Physics-based, data-driven discovery of constitutive equations for granular media

---

Enzo LOUVARD

Thesis

Presented in partial fulfillment of the requirements for the degree of

Master in Civil Engineering

International Program "Geomechanics, Civil Engineering and Risks"

Université Grenoble Alpes

June 2024

Project Advisors:

- Itai EINAV, Professor, The University of Sydney, School of Civil Engineering, SciGEM
- Filippo MASI, Postdoctoral Research Associate, The University of Sydney, School of Civil Engineering, SciGEM

Physics-based, data-driven discovery of constitutive equations for granular media

Enzo Louvard

Abstract

Fueled by a constantly increasing amount of available data, a transformative shift towards data-intensive models – that is, deep learning algorithms – has arisen in recent years. These models demonstrated that it is possible to construct high-fidelity digital replicas of complex materials from large amounts of data sets, or "big data". Yet, most real scenarios, e.g., experiments, require the ability to operate in "small data" regimes – that is, with a sparse and limited number of observations. Despite the promising success of physics-informed deep learning for the discovery of constitutive models in big data regimes, there are only a few approaches capable of learning from small data while also leveraging the universal laws of thermodynamics of continua, i.e., the Neural Integration for Constitutive Equations framework. However, such a physics-based, data-driven approach has never been implemented in learning constitutive equations from real, physical experiments. This work extends the neural integration approach and investigates the possibility of learning from experiments of a heterogeneous, opaque geomaterial, i.e., sand, characterized by complex transient phenomena and multiple inherent scales. In so doing, we showcase the capabilities and performances of the revisited approach in learning from drained triaxial experiments with both monotonous and cyclic loading protocols.

Data and software availability. Code and data pre-/post-processing accompanying this Thesis are publicly available at github.com/enzolvd/NICE_XP.

Acknowledgments

First, I would like to thank Filippo Masi for supervising my project. I appreciate that you always take the time to help me and that you always have the sharpest suggestions. Since my first day, I felt integrated into the project and thought I was part of the progress. It was a pleasure to feel free to take the initiative in a field in which I am not an expert. Overall, Filippo helped me through every aspect of my life in Sydney.

I would also like to thank my co-supervisor, Itai Einav, for bringing his theoretical understanding of geomechanics. Moreover, I would like to thank him for funding my trip to Sydney. This project would not have been possible without those funds.

I am also glad to have met all my colleagues, especially Max, Ilija, and Shivu, with whom I share my breaks. It made my life in Sydney easier.

I am grateful to Professor Cino Viggiani, who always backed us as a teacher and a supervisor and provided us with various offers that could broaden our future careers.

Finally, I would like to thank my family, girlfriend, and friends, who, despite the 10-hour time shift, always supported me and helped me throughout this adventure in Australia.

Contents

List of Figures

List of Tables

Introduction

With increasing data availability and computational power, machine-learning approaches are becoming popular in addressing scientific problems. Machine learning is a sub-field of artificial intelligence that enables computers to learn from data without being explicitly programmed. A machine learning program can work on new data after being given a certain number of examples, referred to as training data [Solomonoff, 1957].

There are many types of machine learning problems, and different solutions are provided. One of the most popular and flexible solutions is referred to as artificial Neural Network (NN). A NN assembles simple cells called neurons. To produce an output, a neuron takes in one or more inputs, assigns weights and a bias, adds up the weighted inputs, and then runs the resultant information through an activation function. Multiple neurons can be assembled to become a layer. Those layers can be stacked to create a neural network. A deep-neural network and deep learning is a specific type of neural network with many layers [Schmidhuber, 2014]. Such neural networks have a wide range of applications. With the rise of big data, those multi-perceptrons have demonstrated a great capacity to construct high-fidelity models (see [Goodfellow *et al.*, 2014, Silver *et al.*, 2016, Radford *et al.*, 2018]).

In mechanics, using machine learning to model the behavior of materials is not new. As soon as 1991, a neural network was trained to capture the behavior of concrete under monotonic biaxial loading and compressive uniaxial cyclic loading [Ghaboussi *et al.*, 1991]. Although the results were promising, the amount of data needed to train and the lack of physical explainability of a neural network were limitations for machine learning to become a viable way of modeling mechanical materials. Data used are extracted from experiments. Therefore, the observations are sparse and limited, and the model has to operate in a "small data" regime. In geomechanics and granular materials, a model to predict soil behavior under cyclic loading was suggested [Gu *et al.*, 2020]. However, the same problems arose. The training relies on a large number of data. Therefore, the model performed well using synthetic data, but the quality dropped with experimental data. Regarding the model's explainability, LSTM networks rely on memory variables that cannot be associated with physical quantities.

To account for the lack of explainability [Ghaboussi *et al.*, 1991, Gu *et al.*, 2020], Physic-Informed Neural Network [Karniadakis *et al.*, 2021] for constitutive modeling can be used. Physic-Informed Machine Learning (PIML) integrates physical

equations like thermodynamics to get physically plausible solutions.

In the context of mechanics, PIML has been used to identify internal variables of a solid [Masi et Stefanou, 2023]. In this framework, the laws of thermodynamics are hardwired within an incremental formulated neural network. With this approach, one step is crossed towards more explainability. However, with an incremental approach, the model requires access to hidden variables, such as elastic strain. From experiments, those hidden variables cannot be accessed. Therefore, a new approach is needed.

The approach from [Masi et Stefanou, 2023], using thermodynamics equations, has been improved by changing the formulation of the problem. The classical mechanic's approach relies on an incremental approach to formulate a model. When using a model formulated incrementally, the state variables, such as the density and the elastic strain, must be known for every step. Even though it is sometimes possible to access those data, it is always delicate, especially for complex materials, such as geomaterials. In [Masi et Einav, 2024], the model is formulated under an integral approach. Using that formulation, the neural network only needs the internal state variables at the initial time. This is a significant step towards learning directly from experiments without assuming any incremental rate for internal variables. To achieve such a formulation, [Masi et Einav, 2024] relies on neural differential equations [Kidger, 2022]. This kind of deep-learning algorithm allows a neural network to learn continuous problems from discrete observations. This framework is the ground layer of the Neural Integration for Constitutive Equations (NICE) approach.

So far, the NICE framework has been benchmarked only on synthetic data. The performances were excellent, even under very noisy conditions. NICE also address the problems in [Ghaboussi *et al.*, 1991, Gu *et al.*, 2020, Masi et Stefanou, 2023]. NICE can learn under a small amount of synthetic data, thus addressing the problem of small data and also overcome the issue of physical explainability.

However, the main central problem of learning from real data is yet to be addressed. In [Gu *et al.*, 2020], the neural network was able to learn from synthetic data. However, turning to real data was a real challenge. This work tries to address this problem. To do so, this work will rely on both the NICE framework and high-fidelity experimental repositories on Karlushe sand [Wichtmann *et al.*, 2015].

This report is structured as follows. Chapter 1 will go through the theoretical background behind [Masi et Einav, 2024] on which our approach is based. Chapter 2 will be the main focus of this work. The datasets and their pre-processing will be showcased, as well as the results extracted from this methodology, their performance, and their accuracy. Finally, Chapter 3 will discuss the framework's capabilities to learn from real small data and the future steps towards better learning from experiences.

The following notation is used: $\mathbf{a}\cdot\mathbf{b} = a_i b_i$, $\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} = \sigma_{ij}\dot{\varepsilon}_{ij}$, $\mathbf{1}$ is the identity tensor, and $\operatorname{div}\mathbf{v} = \partial v_i/\partial x_i$, with x_i the spatial coordinates, v_j a vectorial quantity, and $i, j = 1, 2, 3$. Einstein's summation is implied for repeated indices. For ease of reading, we distinguish functions from their values with a superposed caret, whenever a

marked difference is important: for instance, $\hat{u}$ is a function and u its value.

Chapter 1

Methodology

This chapter will present the Neural Integration for Constitutive Equations (NICE) methodology, as developed in [Masi et Einav, 2024]. The first section will be dedicated to concepts behind machine learning. The second section will be focused on the main framework.

Contents

1 Machine Learning

1.1 Introduction to Machine Learning

Machine Learning (ML) is a sub-field of artificial intelligence that enables computers to learn from data without being explicitly programmed. A machine learning program can work on new data after being given a certain number of examples, referred to as training data [Solomonoff, 1957]. According to Tom Mitchel [Mitchell, 1997], a machine learning program is:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Overall, ML is a powerful tool that can be used to many different ends. In 2016, *AlphaGo* won against the world champion of Go [Silver *et al.*, 2016]. In 2018, OpenAI published the first paper on Generative Pre-trained Transformer (GTP) [Radford *et al.*, 2018], which paved the way for chat-bots as we know them today. The capacities of ML are growing exponentially, as are their complexity. For instance, the number of positions, in Go, from one given initial position is 10^{170} [Tromp et Farnebäck, 2007]. GPT-4, the ML model beneath ChatGPT-4, has over a trillion parameters.

Therefore, using ML in the context of mechanics and material modeling is a logical step forward for predicting and, eventually, understanding the mechanics of complex materials. Yet, there are many challenges in using machine learning in the context of mechanics, such as the amount of available data, the presence of noise, and the underlying physics that have to be respected, etc [Stefanou, 2023]. Before going into the application in geomechanics, we will first set the basis behind machine learning.

1.1.1 Classification of the methods

There are three main types of machine learning: *supervised learning*, *unsupervised learning*, and *reinforcement learning*. Fig.1.1 gives an overview of each type of learning.

Supervised learning
Supervised learning is often associated with regression and classification problems. The goal is to learn the relationship between the data inputs (numbers, pictures, texts, etc.) and corresponding labels [Sebastian Raschka, 2022]. A classical example of supervised learning is classifying pictures from their content (cats, dogs, etc.). Classification corresponds to a context of a discrete, finite number of classes. When the outcome must be continuous, the problem is referred to as a regression problem; for example, fitting a curve over some data points. Fig.1.2 illustrates those two types of supervised ML.

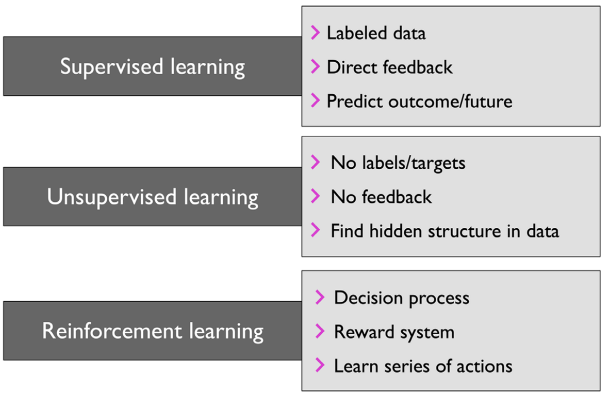

Figure 1.1: The three different types of machine learning, [Sebastian Raschka, 2022].

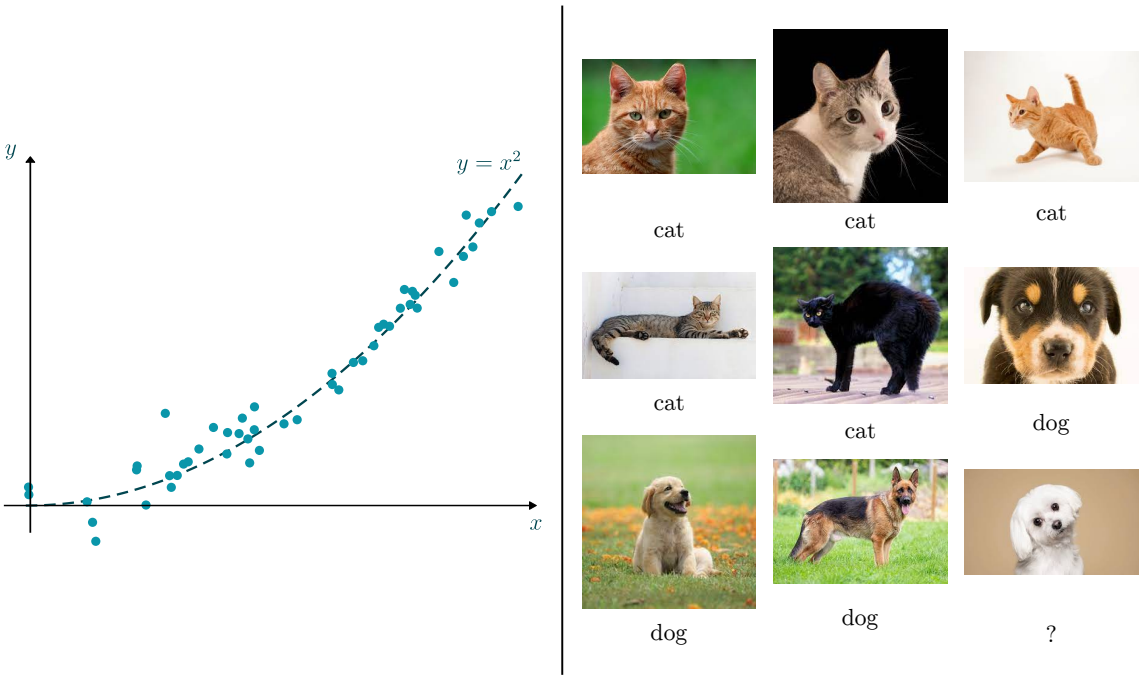

Figure 1.2: Regression problem vs Classification problem.

Unsupervised learning

In the context of unsupervised learning, data do not have a known structure [Sebastian Raschka, 2022]. Unsupervised learning aims to explore the data structure and extract meaningful information. Examples of unsupervised learning are clustering, dimensionality reduction, anomaly detection, etc.

Reinforcement learning

Reinforcement learning aims to create an agent that can improve when interacting with the system [Sebastian Raschka, 2022]. The agent is trained using the

concept of reward. Each time the agent faces the system, it produces an outcome. The agent gets a positive reward if the result is close to the goal. It is used in a lot of daily life applications such as autonomous driving, robotics manipulation,etc

ML can take many forms to solve problems of different natures. For example, it can solve regression problems with a given function shape, minimize a quantity for a clustering problem, and find the best weights and bias for a Neural Network. ML is always about minimizing a quantity, often called a loss function.

1.2 Generality on machine learning

No matter which kind of approach is followed to solve a specific problem, ML always has the same underlying structure. This section delves into the structure of ML. To this end, attention is focused on a regression problem. The goal is to set a function $f_{\boldsymbol{\theta}}(\cdot)$, parameterized by $\boldsymbol{\theta}$, to fit data by solving a minimization problem of a metric error between the predictions and the dependent variables [Masi, 2023a]. The data is defined as follows:

$$\boldsymbol{Y} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{y}^{(1)} & \boldsymbol{y}^{(2)} & \dots & \boldsymbol{y}^{(n)} \\ | & | & | & | \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \dots & \boldsymbol{x}^{(n)} \\ | & | & | & | \end{bmatrix}, \tag{1.1}$$

where $\boldsymbol{x}^{(k)} = [x_1, x_2, \dots, x_m]^T$, and $\boldsymbol{y}^{(k)} = [y_1, y_2, \dots, y_p]^T$

1.2.1 Linear regression

Let us consider a linear regression problem where $f_{\boldsymbol{\theta}}(\cdot)$ is defined by

$$\hat{\boldsymbol{x}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \ldots + \theta_m x_m = \boldsymbol{\theta}^T \boldsymbol{x}^*, \tag{1.2}$$

where $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \dots, \theta_m]^T$ and $\boldsymbol{x}^* = [1, x_1, x_2, \dots, x_m]^T$. To identify the optimal parameters $\hat{\boldsymbol{\theta}}$ such that the function $f_{\boldsymbol{\theta}}(\cdot)$ fits the considered data, training is necessary. The latter is defined as the minimization of the error between target outputs and predictions. Multiple error functions can be defined, with the most commonly adopted being the Mean Square Error (MSE), defined as

$$\text{MSE}(\boldsymbol{X}, f_{\boldsymbol{\theta}}) = \frac{1}{n} \sum_{k=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(k)}) - \boldsymbol{y}^{(k)} \right)^2. \tag{1.3}$$

The goal of the algorithm is to find the optimum $\hat{\boldsymbol{\theta}}$ according to

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}(\text{MSE}(\boldsymbol{X}, f_{\boldsymbol{\theta}})). \tag{1.4}$$

To find a solution to Equation (1.4), $\text{MSE}(\boldsymbol{X}, f_{\boldsymbol{\theta}})$ is differentiated with respect to $\boldsymbol{\theta}$,

$$\frac{\partial \text{MSE}(\boldsymbol{X}, f_{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}} = 0 \tag{1.5}$$

In the linear case, it is possible to find the theoretical solution of (1.4), namely

$$\boldsymbol{X}^T \boldsymbol{X} \hat{\boldsymbol{\theta}}^T = \boldsymbol{X}^T \boldsymbol{Y} \Leftrightarrow \hat{\boldsymbol{\theta}}^T = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{Y} \tag{1.6}$$

1.2.2 Non-linear regression: Gradient descent

If the function is non-linear, it is impossible to find an analytical solution for the optimal parameters. Linear solutions are scarce, non-linear being the norm (see Section 1.3). This section considers a non-linear function $f_{\boldsymbol{\theta}}$. As in the linear case, to find the optimal solution, it is required that $\partial \text{MSE}(\boldsymbol{X}, f_{\boldsymbol{\theta}})/\partial \boldsymbol{\theta} = 0$. This non-linear system is to be solved,

$$\frac{1}{n} \sum_{k=1}^{n} \left(f_{\boldsymbol{\theta}}(x^{(k)}) - \boldsymbol{y}^{(k)} \right) \frac{\partial f_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = 0 \tag{1.7}$$

Gradient Descent (GD) is an iterative optimization algorithm to find a local minimum of a differentiable function [Boyd et Vandenberghe, 2004], being here find the optimal $\boldsymbol{\theta}$ for a given loss function. An intuitive definition is given by [Masi, 2023a]:

> *Imagine yourself lost in a foggy mountain, where visibility is so limited that you can only sense the slope beneath your feet. To find your way out quickly, a smart approach would be to head downhill, following the steepest slope.*

FIG.1.3 illustrate this notion. The concept of gradient descent is the same: a local measure of the gradient with respect to $\boldsymbol{\theta}$ and tweaks this parameter in that direction. It is an iterative process starting from a random value $\boldsymbol{\theta}^{(0)}$ and follows this recursion process,

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \frac{\partial E}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^{(\boldsymbol{k})}, f_{\boldsymbol{\theta}^{(\boldsymbol{k})}}) \tag{1.8}$$

- η is the **learning rate** and is driving the size of the incremental steps. It is a *hyperparameter*, a variable controlling the learning process. Therefore, its identification is crucial to find $\hat{\boldsymbol{\theta}}$. Many iterations will be needed to locate the answer if η is too tiny, making the process time-consuming. Conversely, if η is too large, we might overshoot the solution, resulting in a divergence.

- the **loss function** is also an essential parameter for fast convergence towards the solution. FIG.1.3 illustrates a perfect situation where the loss function

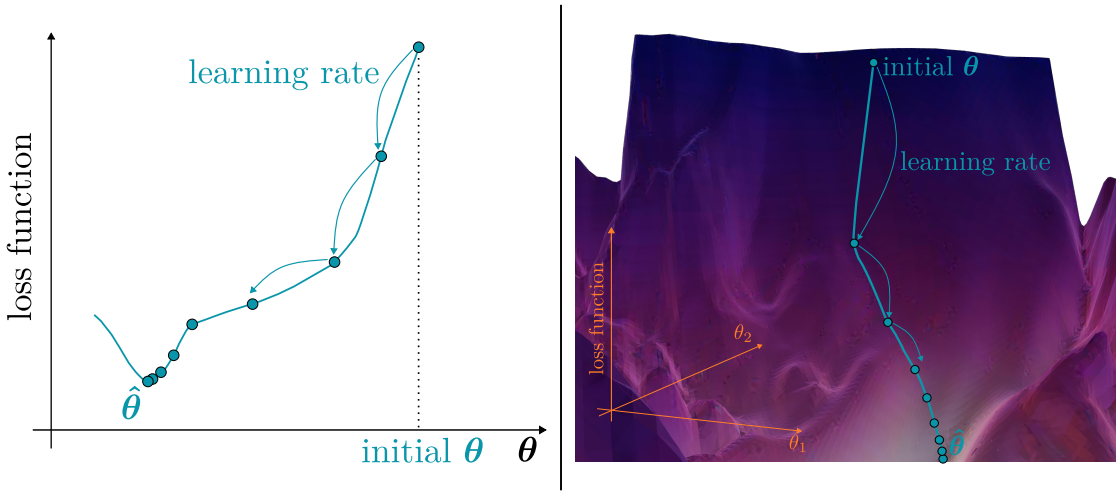

Figure 1.3: Illustration of gradient descent, [Ideami, 2019].

is smooth, with only one minimum. However, the reality is much more complex, with local minimums, plateaus, and points of discontinuity. Therefore, it is crucial to find a function that minimizes those challenges. Also, the loss function must be differentiable knowing (1.8). MSE is the perfect candidate because it is always differentiable, continuous, and convex.

In a 1-D regression problem, the number of parameters is limited. However, when moving on to more complex problems (see Section 1.3), the dimension of the space of solution is increasing exponentially. There are multiple techniques and procedures to perform a gradient descent operation. The major difference relies on how much of the training set you give simultaneously to the algorithm.

Batch Gradient Descent
Batch gradient descent is the easiest but also the heaviest implementation of GD. Equation (1.8) is used with the whole training set $\boldsymbol{X}$ at every step. It is much faster than using Equation (1.6). However, compared to other GD techniques, batch GD is computationally demanding and highly dependent on the initial guess.

Stochastic Gradient Descent
With this method, instead of providing Equation (1.8) with the whole dataset $\boldsymbol{X}$, it only processes a single datum at a time. Over each time step, the loss function is averaged over the number of snapshots. It requires less memory because it only uses a single memory slot at a time. However, Stochastic GD has a less smooth convergence towards $\hat{\boldsymbol{\theta}}$ because of its nature.

Mini-Batch Gradient Descent
This method lies in between Stochastic GD and Batch GD. Instead of taking all

the snapshots or only one, mini-batch GD uses small random sets of data points known as *mini-batches*. This approach offers both the advantages of Batch GD and Stochastic GD: smoother convergence and reduced memory usage during training. The size of the mini-batches gives the balance between the two previous approaches.

FIG.1.4 illustrates the 3 methods explained above. We can see that Batch GD is super-smooth but super slow. Stochastic GD is much faster but erratic. Mini-batch GD is a combination of both of them, with fast convergence and less erratic behavior.

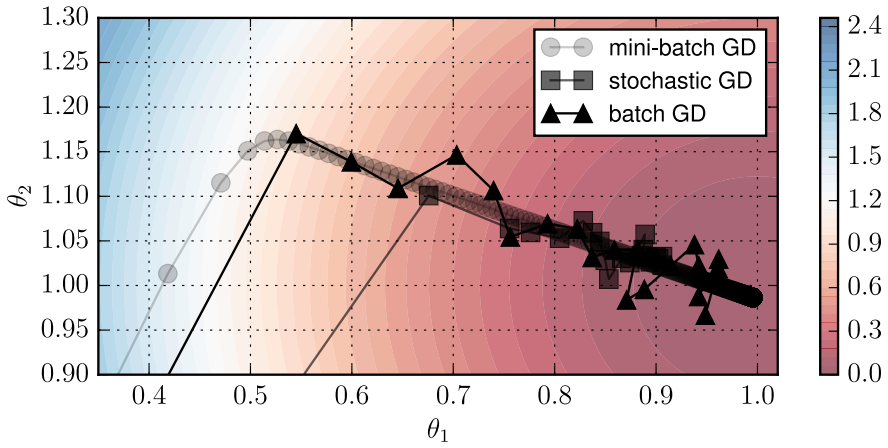

Figure 1.4: Comparison of GD methods, [Masi, 2023a].

1.2.3 Data pre-processing

As explained in the previous sections, machine learning needs data to work. First, they are used as training data to find the best set of parameters for the model. This period is referred to as the training phase. However, if all the data are used as training values, the algorithm will most probably overfit the data [Masi, 2023a]:

> *Overfitting is defined as the phenomenon where a model becomes overly complex and excessively fits the training data set, capturing noise and irrelevant patterns instead of learning the true underlying patterns and relationships*

To avoid that problem, the solution is to divide the data into sets: training set to use as training and validation set to check during training that the algorithm is not overfitting (see Section 1.2.4. This method is referred to as train-validation-test splitting [Joseph, 2022].

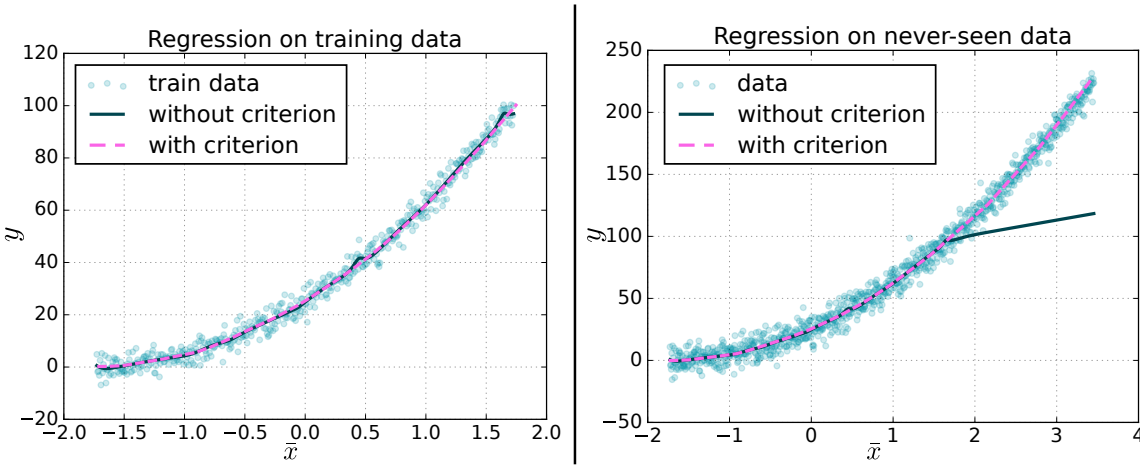

Figure 1.5: Regression on (i) training data and (ii) never-seen data, with and without a stopping criterion.

1.2.4 Stopping criterion

Stopping criteria are crucial to performing training without overfitting. There are many ways of choosing it. It is, for instance, possible to define a patience period, which, if for that patience period, the validation loss does not improve, then the training stops. Another criterion class relies on a given parameter that the validation loss should reach [Prechelt, 1998].

Fig.1.5 showcases the importance of a good early stopping criterion. Without the latter, the model largely focuses on the noise of the data in the training (i), and fails in extrapolating. On the other hand, with a stopping criterion, in the training phase the model captures the trend and does not focus on the noise. This also results in higher performances at extrapolating results.

1.2.5 Overview of the process

Fig.1.6 gives an overview of the learning process in machine learning frameworks.

1.3 Neural Network

Artificial Neural Network, or for short Neural Network, is a technique used in ML to model large panels of problems. Neural networks are interconnected artificial neurons that work together to perform complex tasks. The artificial neuron is the basic building block of a neural network.

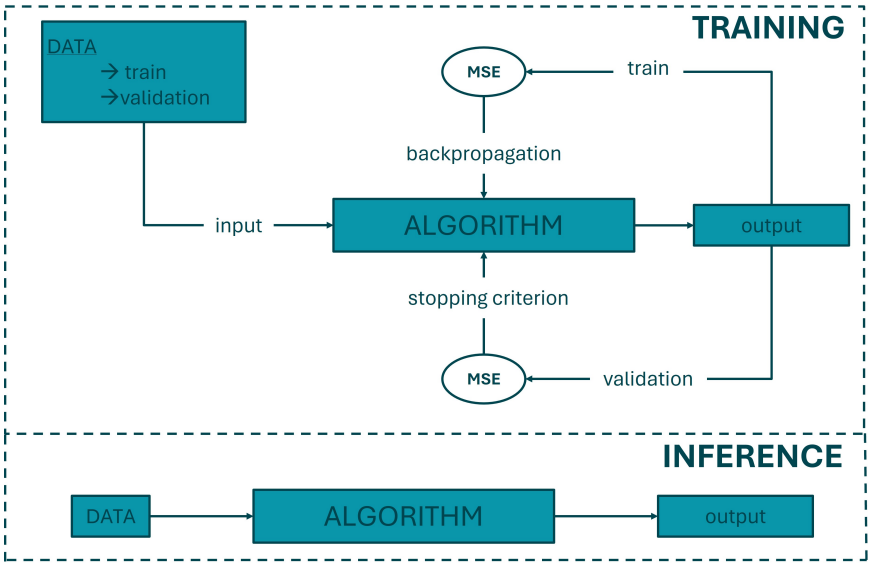

Figure 1.6: Overview of ML.

1.3.1 Artificial neuron

Historical definition

Artificial neurons, first defined in 1957 [Rosenblatt, 1957], are based on the biological neuron. A cell is provided with an input signal (data for a perceptron and electrical or chemical signal) to process it and provide an output signal. A biological neuron is as simple as a logic gate, an "all-or-none" unit that fires action potentials when its excitation exceeds a certain threshold and then propagates these impulses through its axonal connections to other neurons [McCulloch et Pitts, 1943]. In [Rosenblatt, 1957], the authors develop an algorithm to find the ideal weight coefficients to multiply the input information to determine whether a neuron transmits a signal. For the problem of machine learning, especially supervised learning, this algorithm can be used, for example, to predict a class in a classification problem [Sebastian Raschka, 2022].

Mathematical formulation

A neuron performs this non-linear transformation:

$$h_\theta(\boldsymbol{x}) = g(a(\boldsymbol{x})) = g(\boldsymbol{w}^T \cdot \boldsymbol{x} + b); \quad \boldsymbol{\theta} := \{\boldsymbol{w}; b\} \tag{1.9}$$

a is referred to as linear *pre-activation*. $\boldsymbol{w}$ and b are the parameters of the neuron, referred to as weights and bias. g is the activation function.

Activation function: The activation function is the part of the neuron that performs the non-linear transformation. Fig.1.7 gives an overview of the classic

activation functions:

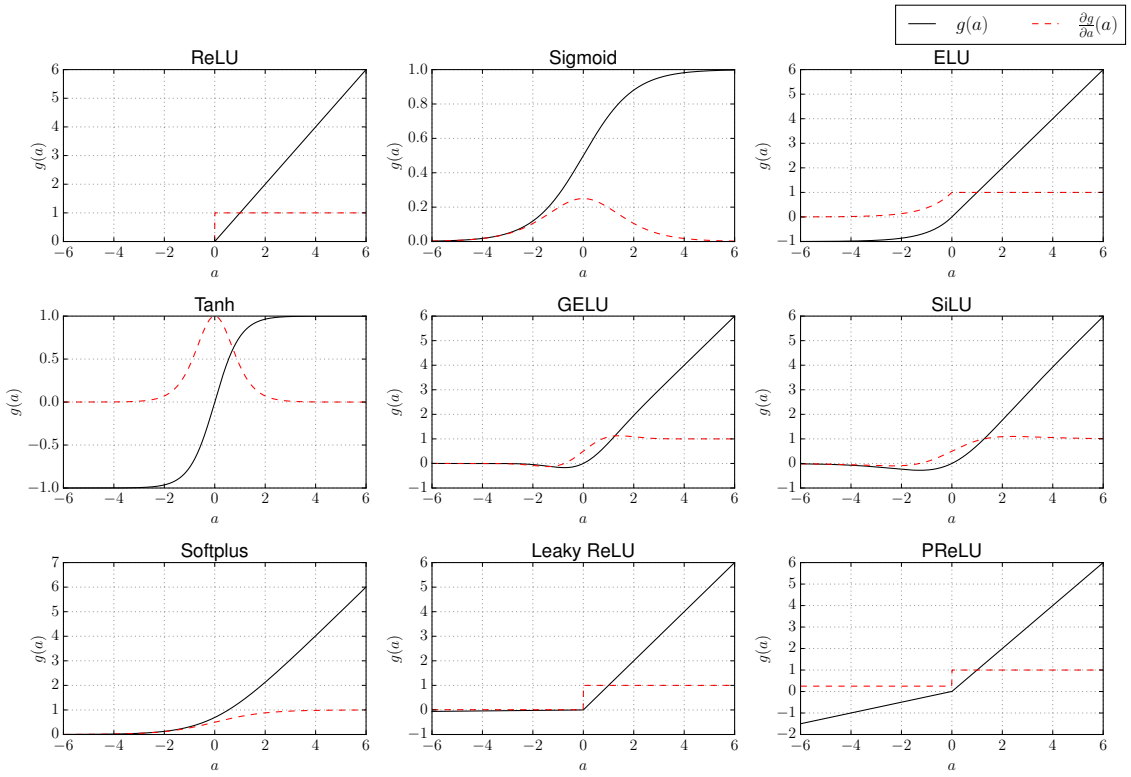

Figure 1.7: Classic activation functions in ML, [Gatti, 2023].

1.3.2 Multi-Layer Perceptron

A Multi-Layer Perceptron ($\mathcal{MLP}$) is an object where multiple layers of neurons are stacked upon each other. A single neuron is insufficient for problems where the output is not linear (for instance, a classification problem with a non-linear boundary line). For example, adding one layer, called a hidden layer, drastically raises the complexity of the model. The general case is described in Fig.1.8. This figure showcases the connection between the layer l and $l-1$. Generally, a $\mathcal{MLP}$ gathers multiple layers, with each neuron c in the layer l, being fed by the layer $l-1$ and outputting data into the layer $l+1$. For each neuron, the following notation is taken,

- weight vector $\boldsymbol{w}_c$
- bias b_c

The weight vectors can be stacked into a matrix $\boldsymbol{W}$ for each layer. However, the notion of is flexible; a fully connected MLP has to follow those rules:

- activation function is the same for neurons on a given layer
- neurons for a given layer receive the same input

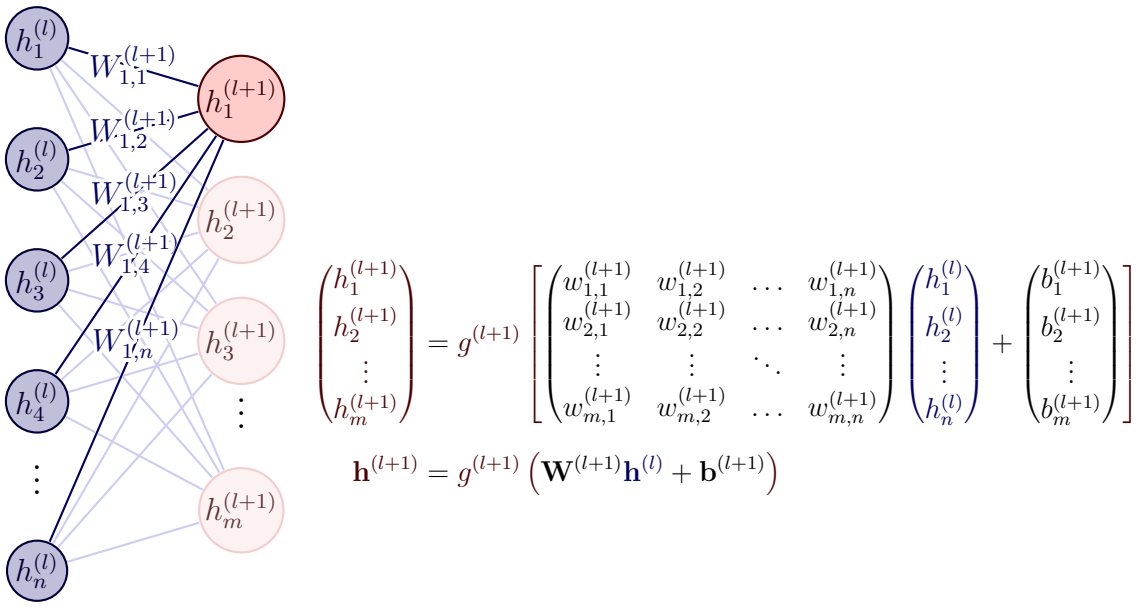

Figure 1.8: Activation of a fully connected, $\mathcal{MLP}$ [Gatti, 2023].

To sum up, a is a non-linear function, with weights and bias, that can evolve :

$$h_{\boldsymbol{\theta}} : \boldsymbol{x} \longmapsto h_{\boldsymbol{\theta}_L} \circ h_{\boldsymbol{\theta}_{L-1}} \circ \ldots \circ h_{\boldsymbol{\theta}_1} \circ h_{\boldsymbol{\theta}_0}(\boldsymbol{x}) \tag{1.10}$$

$$h_{\boldsymbol{\theta}_l} : \boldsymbol{x^{(l-1)}} \longmapsto g^{(l)} \left(\boldsymbol{W^l} x^{(l-1)} + b^l \right) \tag{1.11}$$

The weights and biases are chosen through a training phase, described in Section 1.2.2

2 Machine learning for constitutive equations

As exposed in the introduction, ML is used in the context of modeling materials and geomaterials [Ghaboussi *et al.*, 1991, Gu *et al.*, 2020, Masi et Stefanou, 2023, Masi et Einav, 2024]. The latter use several different approaches to discover and learn constitutive equations. Here, we leverage the Neural Integration for Constitutive Equations (NICE) framework, [Masi et Einav, 2024]. NICE relies on two main features:

- Physic Informed Machine Learning: the laws of thermodynamics are directly hard-wired within the architecture of neural networks [Karniadakis *et al.*, 2021],

- Neural Differential Equation: the constitutive equations are formulated according to an initial value problem, where only initial conditions for the state variables are required [Kidger, 2022].

In the following, we present the aforementioned frameworks.

2.1 Physic Informed Machine Learning

PIML or PINN (Physics Informed Neural Network) is a class of ML where physical laws can be encoded inside the Network. For a given non-linear Partial Differential Equation (PDE),

$$\dot{\boldsymbol{u}} + \mathcal{N}_\gamma[\boldsymbol{u}] = 0, \tag{1.12}$$

where $\boldsymbol{u}$ is the solution to be found. $\mathcal{N}_\gamma$ is a non-linear differential operator. $\boldsymbol{u}$ is model by a deep neural network $\boldsymbol{u_\theta}$. Therefore (1.12) is written as,

$$\mathcal{Q}(\boldsymbol{x}, t) = \dot{\boldsymbol{u}}_{\boldsymbol{\theta}} + \mathcal{N}[\boldsymbol{u}_{\boldsymbol{\theta}}]. \tag{1.13}$$

The loss function includes a loss over data measurement as well as a loss over PDE:

$$\mathcal{L} = \lambda_{\boldsymbol{u}}\mathcal{L}_{\boldsymbol{u}} + \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} \tag{1.14}$$

$$= \frac{\lambda_{\boldsymbol{u}}}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \left\| \boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, t^{(i)}) - \boldsymbol{u}^{(i)} \right\| + \frac{\lambda_{\text{PDE}}}{N_{\text{PDE}}} \sum_{k=1}^{N_{\text{PDE}}} \left\| \mathcal{Q}(\boldsymbol{x}^{(k)}, t^{(k)}) \right\|. \tag{1.15}$$

$\mathcal{L}_{\text{PDE}}$ is referred to as learning bias. Other types of bias can be implemented, such as observational or inductive (see [Karniadakis *et al.*, 2021]). The algorithm can be summed up with Fig.1.9

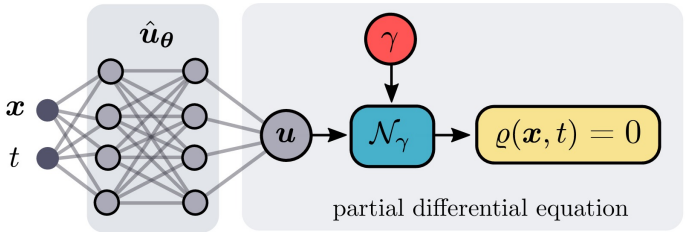

Figure 1.9: PINN, [Masi, 2023b].

2.2 Neural Differential Equations

2.2.1 Incremental formulation

Most data-driven deep learning approaches for constitutive modeling rely on *incremental formulation* (see [Ghaboussi *et al.*, 1991, Gu *et al.*, 2020, Masi et Stefanou, 2023]). At each time step, a state $\mathbf{y}$ of a given time-evolving system follows,

$$\mathbf{y}^{(n)} = \mathbf{y}^{(n-1)} + \boldsymbol{h}_\theta(\mathbf{y}^{(n-1)}, \mathbf{q}), \tag{1.16}$$

where $\mathbf{y}^{(n)}$ is the value of the state at discrete times $\{t^{(n)}\}_{n=1}^{N}$; $\boldsymbol{h_\theta}$ a generic neural operator that describes the evolution of the state; and $\mathbf{q}$ are control variables, for instance time.
Training $\boldsymbol{h_\theta}$ requires dense, regularly sampled data that describe the entire state at each time step. When only partial information at a time $t^{(n)}$, inferring the next time step is impossible. This is the major limitation of this approach.

2.2.2 Integral formulation

Neural Differential Equations (NDE) are deep learning algorithms that allow the parameterization of time-continuous states formulated as ODE,

$$\dot{\mathbf{y}} = \frac{d\mathbf{y}}{dt}(t) = \boldsymbol{f_\theta}(\mathbf{y}(t), \mathbf{q}), \tag{1.17}$$

where $\mathbf{q}$ are the control variables and $\boldsymbol{f_\theta}$ is an evolution equation parametrized by a feed-forward neural network that has to be integrable.
From the knowledge of $\mathbf{y}^{(0)}$, the state at any time is obtained by solving an initial value problem,

$$\mathbf{y}_\theta(t) = \mathbf{y}^{(0)} + \int_0^t \boldsymbol{f_\theta}(\mathbf{y}(\tau), \mathbf{q}) d\tau = \underset{[0,t]}{\text{ODESolve}}(\mathbf{y}^{(0)}, \boldsymbol{f_\theta}, \mathbf{q}) \tag{1.18}$$

The parameters $\boldsymbol{\theta}$ are learned by a gradient descent algorithm (see 1.2.2). Using, for example, MSE, the goal is to minimize,

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \left(\underset{[0,t^{(n)}]}{\text{ODESolve}}(\mathbf{y}^{(0)}, \boldsymbol{f_\theta}, \mathbf{q}) - \mathbf{y}^{(n)} \right), \tag{1.19}$$

where $t = t^{(1)}, \ldots, t^{(N)}$ are discrete time steps.

2.3 Formulation in NICE

2.3.1 Overview

Let us explore NICE and its formulation. Fig.1.10, extracted from [Masi et Einav, 2024] is a global overview of NICE,

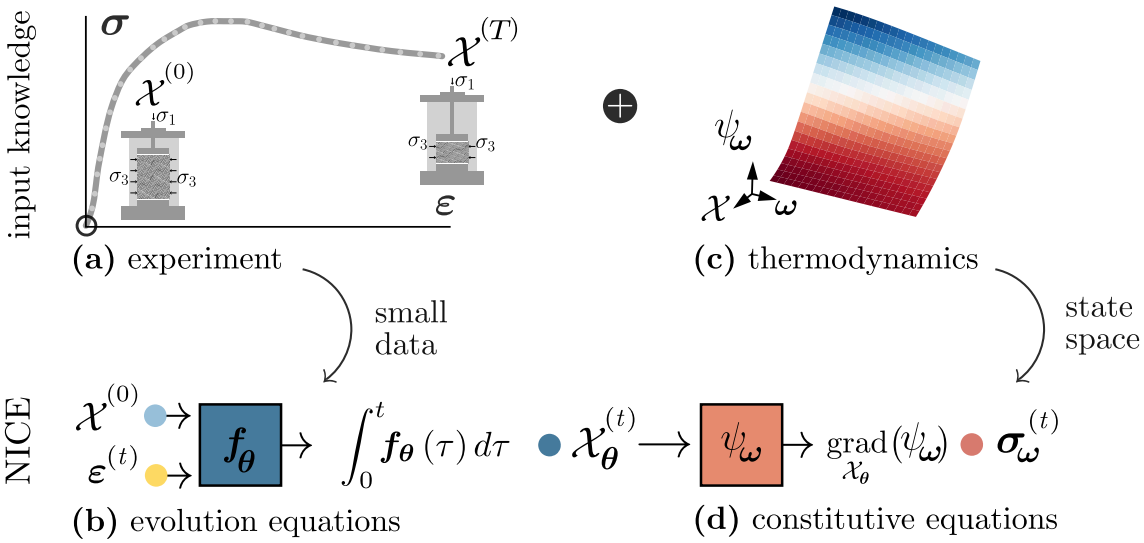

Figure 1.10: Graphical illustration of NICE: Neural Integration for Constitutive Equations, [Masi et Einav, 2024].

(a) provide the stress-strain response from experimental tests as well as the initial and final incomplete material configuration described by the state variables $\boldsymbol{\chi}^{(0)}$ and $\boldsymbol{\chi}^{(T)}$. From (a), the framework learns the evolution equations (b) through a neural differential operator $\boldsymbol{f_\theta}$, parametrized by parameters $\boldsymbol{\theta}$. $\boldsymbol{f_\theta}$ is then integrated to compute the state space trajectory $\boldsymbol{\chi}^t_{\boldsymbol{\theta}}$. Then, the stress is computed (d), using a second network representing the specific free energy $\psi_{\boldsymbol{\omega}}$ along with constitutive constraints extracted from thermodynamic principles (c).

2.3.2 Thermodynamics and constitutive restrictions

From [Masi et Einav, 2024], the thermodynamics formulation has been improved. A porous, single-component material is considered. It is undergoing infinitesimal strains and isothermal processes. The state of the material is assumed to be completely defined by a set of $p+2$ state variables $\boldsymbol{\chi} = \{\rho, \boldsymbol{\varepsilon}^e, z_1, \ldots, z_p\}$, where ρ is the volumetric mass density, $\boldsymbol{\varepsilon}^e$ is the elastic strain, and z_i are optional dissipative state variables describing irreversible phenomena. The notation used is $\boldsymbol{z} = \{z_1, \ldots, z_p\}$. Therefore, using the axiom of equipresence, the Helmholtz free-energy density is defined as :

$$\psi \equiv \hat{\psi}(\rho, \boldsymbol{\varepsilon}^e, \boldsymbol{z}). \tag{1.20}$$

Constitutive equations must comply with the energy balance (first law of thermodynamics) and the entropy inequality (second law of thermodynamics). The combination of the latter is given by the (Clausius-Duhem) dissipation inequality, namely

$$d = \boldsymbol{\sigma} : \boldsymbol{\varepsilon^e} - \rho\dot{\psi} \geq 0, \tag{1.21}$$

where d is the specific mechanical dissipation rate. Coming from (1.20), the time derivative of the free energy is

$$\dot{\psi} = \frac{\partial\hat{\psi}}{\partial\rho}\dot{\rho} + \frac{\partial\hat{\psi}}{\partial\varepsilon^e} : \dot{\boldsymbol{\varepsilon}}^e + \frac{\partial\hat{\psi}}{\partial\boldsymbol{z}} \cdot \dot{\boldsymbol{z}}. \tag{1.22}$$

The plastic strain rate $\dot{\boldsymbol{\varepsilon}}^p$ is also defined,

$$\dot{\boldsymbol{\varepsilon}}^p \equiv \dot{\boldsymbol{\varepsilon}} - \dot{\boldsymbol{\varepsilon}}^e. \tag{1.23}$$

From the mass balance equation, $\dot{\rho} + \rho \operatorname{div} \mathbf{v} = 0$, knowing that $\mathbf{v} = \frac{d\mathbf{x}}{dt}$, the evolution of the volumetric mass density follows,

$$\frac{\dot{\rho}}{\rho} = \mathbf{1} : \dot{\boldsymbol{\varepsilon}}^e. \tag{1.24}$$

Finally using Equations (1.21-1.24), this equation is obtained:

$$\left(\boldsymbol{\sigma} - \rho\left(\frac{\partial\hat{\psi}}{\partial\varepsilon^e} + \rho\frac{\partial\hat{\psi}}{\partial\rho}\mathbf{1}\right)\right) : \dot{\boldsymbol{\varepsilon}}^e + \left(\boldsymbol{\sigma} - \rho^2\frac{\partial\hat{\psi}}{\partial\rho}\mathbf{1}\right) : \dot{\boldsymbol{\varepsilon}}^p - \rho\frac{\partial\hat{\psi}}{\partial\boldsymbol{z}} \cdot \dot{\boldsymbol{z}} - d = 0. \tag{1.25}$$

Considering that the dissipation rate does not depend on $\dot{\boldsymbol{\varepsilon}}^e$ leads to the following constitutive restrictions:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^e + p^T\mathbf{1}, \tag{1.26a}$$

$$d = \boldsymbol{\sigma}^e : \dot{\boldsymbol{\varepsilon}}^p + \boldsymbol{\tau} \cdot \dot{\boldsymbol{z}}, \tag{1.26b}$$

$$\boldsymbol{\sigma}^e \equiv \rho\frac{\partial\hat{\psi}}{\partial\boldsymbol{\varepsilon}^e}, \quad p^T \equiv \rho\mu, \quad \mu \equiv \rho\frac{\partial\hat{\psi}}{\partial\rho}, \quad \boldsymbol{\tau} \equiv -\rho\frac{\partial\hat{\psi}}{\partial\boldsymbol{z}}, \tag{1.26c}$$

where $\boldsymbol{\sigma}^e$ is the elastic stress, conjugate to $\boldsymbol{\varepsilon}^e$; μ is the chemical potential, conjugate to ρ; $\boldsymbol{\tau}$ are the thermodynamic forces of the dissipative mechanisms, conjugate to $\boldsymbol{z}$; and p^T is the thermodynamic pressure.
Constitutive equations (1.26) are accompanied by the evolution equations of the state variable, given by,

$$\dot{\boldsymbol{\chi}} = \boldsymbol{f}(\boldsymbol{\chi}, \dot{\boldsymbol{\varepsilon}}), \quad \boldsymbol{f} = \{\boldsymbol{f}^\rho, \boldsymbol{f}^{\boldsymbol{\varepsilon}^e}, \boldsymbol{f}^{\boldsymbol{z}}\} \tag{1.27}$$

Equation (1.27) has to fulfill Equation (1.21), and $\boldsymbol{f}^\rho = \rho(\mathbf{1} : \dot{\boldsymbol{\varepsilon}})$ from Equation (1.24).

2.3.3 Neural Integration for Constitutive Equations

Built upon Section 2.2 and 2.3.2, NICE was a new approach to discovering constitutive equations in small data regimes. It has the advantage of not requiring a dense sampling of state space but also not requiring a complete measurement of the data. Moreover, the time stepping can be variable.
To achieve this goal, we want to identify both $\boldsymbol{f}(\boldsymbol{\chi}, \dot{\boldsymbol{\varepsilon}})$ and $\hat{\psi}(\boldsymbol{\chi})$, that needs to satisfy thermodynamic principles. To do so, both the unknown functions are replaced by neural operators, $\boldsymbol{f_\theta}(\boldsymbol{\chi}, \dot{\boldsymbol{\varepsilon}})$ and $\psi_{\boldsymbol{\omega}}(\boldsymbol{\chi})$. Equations (1.26) are rewritten using neural operators,

$$\boldsymbol{\chi_\theta}(t) = \underset{[0,t]}{\text{ODESolve}}(\boldsymbol{\chi}^{(0)}, \boldsymbol{f}_\theta, \dot{\boldsymbol{\varepsilon}}), \tag{1.28a}$$

$$\boldsymbol{\sigma_{\theta\omega}}(t) = \rho_{\boldsymbol{\theta}} \frac{\partial \psi_\omega}{\partial \boldsymbol{\varepsilon}^e_{\boldsymbol{\theta}}} + \rho^2_{\boldsymbol{\theta}} \frac{\partial \psi_\omega}{\partial \rho_{\boldsymbol{\theta}}} \mathbf{1}, \tag{1.28b}$$

$$d_{\boldsymbol{\theta\omega}}(t) = \rho_{\boldsymbol{\theta}} \frac{\partial \psi_\omega}{\partial \boldsymbol{\varepsilon}^e_{\boldsymbol{\theta}}} : (\dot{\boldsymbol{\varepsilon}} - \boldsymbol{f}^{\boldsymbol{\varepsilon}^e}_{\boldsymbol{\theta}}) - \rho_{\boldsymbol{\theta}} \frac{\partial u_\omega}{\partial \boldsymbol{z}_{\boldsymbol{\theta}}} \cdot \boldsymbol{f}^{\boldsymbol{z}}_{\boldsymbol{\theta}} \geq 0. \tag{1.28c}$$

Equation (1.28a) allows inferring the values of state space variables at any time from the initial values. The other equations from (1.28) originate from (1.26), where the neural operators have replaced the unknown functions.

Initial conditions

The framework must be fed with initial values to predict the evolution of the state space variables. Although it is possible to have access to the density or the other optional variables, it is impossible to measure the elastic strain. Classic approaches suggest giving a prior value for the elastic strain or giving a form to the internal energy and finding the elastic strain. Those two approaches constrain material behavior, which may conflict with thermodynamics.
To identify this unknown variable, the algorithm will guess the initial value using the knowledge of the other known initial state as well as the stress and solve this non-linear equation,

$$\text{find } \boldsymbol{\varepsilon}^{e(0)} \text{ such that } \boldsymbol{r}^0_{\boldsymbol{\varepsilon}^e} = \boldsymbol{\sigma_{\theta\omega}}(0) - \boldsymbol{\sigma}^{(0)} = 0 \tag{1.29}$$

Instead of searching the solution at each epoch of (1.29), $\boldsymbol{\varepsilon^{e(0)}}$ is treated as a hyperparameter, initialized at zero and evolving using gradient descent by back-propagating $\boldsymbol{r^0_{\varepsilon^e}}$.

Learning process

The learning process of the algorithm minimizes this loss function,

$$\mathcal{L} = \mathcal{L}_{\boldsymbol{\sigma}} + \mathcal{L}_{\boldsymbol{z}} + \mathcal{L}_d + \left\| \boldsymbol{r}^0_{\boldsymbol{\varepsilon}^e} \right\|^2_2 + \lambda \left(\|\boldsymbol{\theta}\|_2 + \|\boldsymbol{\omega}\|_2 \right). \tag{1.30}$$

$\mathcal{L}_{\boldsymbol{\sigma}}$ and $\mathcal{L}_{\boldsymbol{z}}$ are supervised losses related to the prediction of the stress and the state variables. $\mathcal{L}_{\boldsymbol{d}}$ is an unsupervised loss to respect the dissipation inequality. $\boldsymbol{r}^{\mathbf{0}}_{\boldsymbol{\varepsilon}^e}$ represent the initial elastic strain problem. Finally $\|\cdot\|_{\mathbf{2}}$ represents the l_2 norm and λ the parameter controlling the importance of the l_2 regularization of $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$. The loss functions are described below:

$$\mathcal{L}_{\boldsymbol{\sigma}} = \frac{1}{N}\sum_{n=1}^{N}\left(\boldsymbol{\sigma}_{\boldsymbol{\theta\omega}}\left(t^n\right) - \boldsymbol{\sigma}^{(n)}\right)^2, \tag{1.31a}$$

$$\mathcal{L}_{\boldsymbol{z}} = \frac{1}{M}\sum_{m=1}^{M}\left(\underset{(0,t^m]}{\text{ODESolve}}\left(\boldsymbol{z}^{(0)}, \boldsymbol{f}^{\boldsymbol{z}}_{\boldsymbol{\theta}}, \dot{\boldsymbol{\varepsilon}}\right) - \boldsymbol{z}^{(m)}\right)^2, \tag{1.31b}$$

$$\mathcal{L}_{d} = \frac{1}{N}\sum_{n=1}^{N}\left\langle -d_{\boldsymbol{\theta\omega}}\left(t^n\right)\right\rangle. \tag{1.31c}$$

t^n denotes the sampling time associated with stress-strain measurements, and t^m specifies the time steps related to state-space variables measurements. For small data regimes, $M \ll N$ and $t^N = t^M = T$ represent the last time step.

Neural Network Restriction

Theoretical reason: We want to constraint by construction the NN computing the energy. Fig.1.11 illustrates what the neural Network has to satisfy.

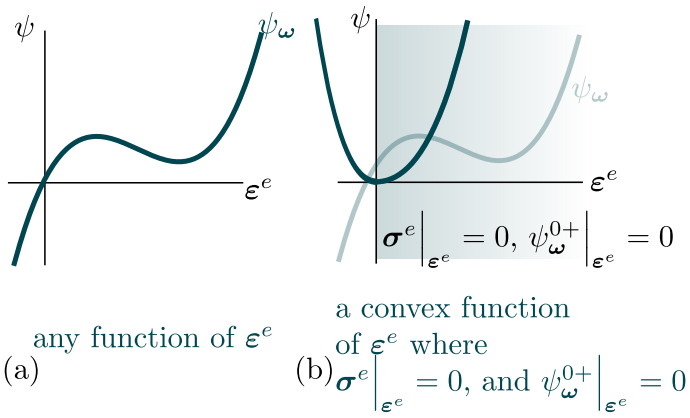

Figure 1.11: Graphical illustration of the energy requirement.

In [Masi et Einav, 2024], the energy neural network is a function without any specific convexity properties, (a) in Fig.1.11. With the experimental approach, we used theoretical knowledge to refine this energy function. First, the energy function has to be convex with respect to $\boldsymbol{\varepsilon}^e$. This convexity is limited to the zone where the material is stable. Then, to have zero energy at zero elastic strain, the energy

is constrained to be zero at zero elastic strain. We also impose zero elastic stress at zero elastic strain by imposing

$$\boldsymbol{\sigma}^e\Big|_{\boldsymbol{\varepsilon}^e=0} = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}^e}\Big|_{\boldsymbol{\varepsilon}^e=0} = 0, \tag{1.32}$$

(a) in FIG.1.11 illustrate what the energy function should look like.

Implementation We use Input Convex Neural Networks (ICNN) [Klein *et al.*, 2022] to respect the constraints developed in the previous section. Even though the Neural Network only needs to be convex with respect to the elastic strain, it is easier to implement convexity with respect to all the inputs, here being the elastic strain, the density, and any other state variable. This is the main objective of ICNN. From any FNN, ICNNs are constructed by applying the following restrictions:

- the first hidden layer $\boldsymbol{h}^{(1)}$ has to be neuron-wise convex with respect to the input: it can be done by using convex activation functions, such as softplus (see FIG.1.7),
- from the second layer $\boldsymbol{h}^{(2)}$ to the last hidden layer $\boldsymbol{h}^{(H)}$ each layer has to be neuron-wise convex and non-decreasing with respect to the previously hidden layer: it is done by using softplus and non-negative weights: as it is linear, it is convex and non-decreasing if the weights of the output layer are positive,
- the output layer is convex and non-decreasing with respect to the last hidden layer,
- if the bias of the output layer is positive, then, given the above conditions, the scalar-valued output is positive and convex.

These conditions allow the NN to be convex with respect to the input (b) in FIG.1.11.

Chapter 2

Results

This chapter delves into exploiting the methodology presented in Chapter 1. The first two sections explore the nature of the considered experimental data and their preprocessing. The third section examines the training procedure and showcases the obtained results. The last section explores the prediction made on unobserved experimental data.

Contents

1 Experimental data collection

Data used in this work are extracted from the online database built by Witchmann in [Wichtmann *et al.*, 2015]. This section describes the experimental procedures used to obtain those data. The material tested is "*Karlsruhe fine sand*". FIG.2.1 displays the grain size distribution, with a mean grain diameter $d_{50} = 0.14$ mm and a uniformity coefficient $C_u = d_{60}/d_{10} = 1.5$. The minimum and maximum values of the void ratio are $e_{\min} = 0.667$ and $e_{\max} = 1.054$, respectively.

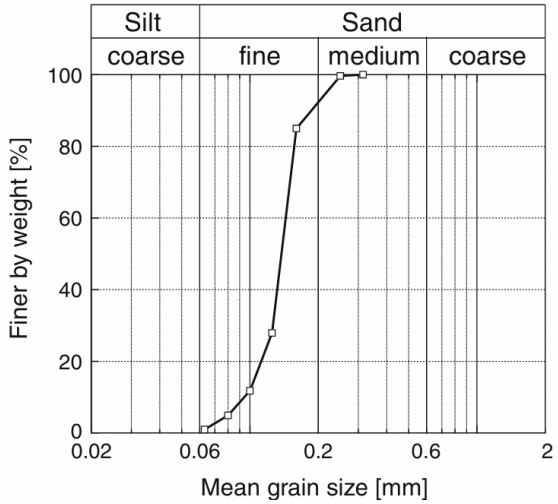

Figure 2.1: Tested grain size distribution curve, [Wichtmann *et al.*, 2015].

Throughout this work, we formulate the material response according to a triaxial formulation – that is, by means of the stress-strain volumetric and deviatoric components: $\varepsilon_v, \varepsilon_s$ and p, q. More precisely, $\varepsilon_v = \mathbf{1} : \boldsymbol{\varepsilon}$ is the volumetric invariant of the strain tensor; $\varepsilon_s = \sqrt{2/3\ \boldsymbol{\varepsilon}' : \boldsymbol{\varepsilon}'}$ is the invariant of the deviatoric strain tensor, where $\boldsymbol{\varepsilon}' = \boldsymbol{\varepsilon} - \frac{\varepsilon_v}{3}\mathbf{1}$; $p = \mathbf{1} : \frac{\boldsymbol{\sigma}}{3}$ is the volumetric stress and $q = \sqrt{3/2\ \boldsymbol{s} : \boldsymbol{s}}$ is the invariant of the deviatoric stress, where $\boldsymbol{s} = \boldsymbol{\sigma} - p\mathbf{1}$. In addition, we compute the volumetric mass density ρ from the void ratio which is provided in the database, assuming a grain density $\rho_s = 2.65 g/cm^3$ [Poblete *et al.*, 2016], i.e.,

$$\rho = \frac{\rho_s}{1 + e_0}. \tag{2.1}$$

1.1 Triaxial Monotonic tests

1.1.1 Drained Triaxial test

Those tests have been performed on a diameter cylinder $d \approx 100$ mm and a height of $h \approx 100$ mm. Twenty-five monotonic Drained Triaxial Monotonic compression tests are performed, referred to as TMD, with different initial relative densities $0.15 \leq I_{D0} \leq 0.95$ where

$$I_{D0} = \frac{e_{\max} - e_0}{e_{\max} - e_{\min}}. \tag{2.2}$$

The samples are consolidated isotropically before the start of each test. Each density group considers five different effective confining pressures, namely $\sigma_3^{'} = \sigma_2^{'} = 50, 100, 200, 300, 400$ kPa. The shearing rate is 0.1 mm/min. Table 2.1 resumes the initial conditions and labeling of the TMD protocols.

Test	e_0 (-)	I_{D0} (-)	p_0 (kPa)
TMD1	0.996	0.15	50
TMD2	0.975	0.21	100
TMD3	0.975	0.21	200
TMD4	0.970	0.22	300
TMD5	0.960	0.25	400
TMD6	0.880	0.46	50
TMD7	0.862	0.51	100
TMD8	0.859	0.52	200
TMD9	0.848	0.55	300
TMD10	0.847	0.55	400
TMD11	0.840	0.57	50
TMD12	0.819	0.63	100
TMD13	0.824	0.63	200
TMD14	0.822	0.64	300
TMD15	0.814	0.68	400
TMD16	0.743	0.82	50
TMD17	0.758	0.79	100
TMD18	0.748	0.81	200
TMD19	0.734	0.85	300
TMD20	0.753	0.80	400
TMD21	0.734	0.85	50
TMD22	0.735	0.85	100
TMD23	0.706	0.92	200
TMD24	0.697	0.95	300
TMD25	0.718	0.89	400

Table 2.1: Program of TMD, [Wichtmann *et al.*, 2015].

We can access several quantities for TMD in the database, including $\varepsilon_1, \varepsilon_v, p, q, e$.

1.1.2 Undrained Triaxial test

In [Wichtmann *et al.*, 2015], two series of undrained triaxial tests exist. For this report, only the first one is considered. In this series, undrained triaxial tests,

referred to as TMU, are performed under initial volumetric stress ranging from 100 kPa to 400 kPa and relative density ranging between 0.24 and 0.94. The shearing rate is 0.02 mm/min. Six of the twelve loading paths are in compression, while the remaining ones are in extension. Table 2.2 resumes the initial conditions and labeling of the TMU protocols.

Test	e_0 (-)	I_{D0} (-)	p_0 (kPa)	Compression/Extension
TMU1	0.828	0.60	100	Comp
TMU2	0.814	0.64	200	Comp
TMU3	0.822	0.62	300	Comp
TMU4	0.819	0.62	400	Comp
TMU5	0.946	0.29	200	Comp
TMU6	0.728	0.87	200	Comp
TMU7	0.828	0.60	100	Ext
TMU8	0.853	0.53	200	Ext
TMU9	0.828	0.60	300	Ext
TMU10	0.827	0.60	400	Ext
TMU11	0.964	0.24	200	Ext
TMU12	0.698	0.94	200	Ext

Table 2.2: Program of TMU, [Wichtmann *et al.*, 2015].

We can access several quantities for TMU in the database, including ε_1, p, q, e_0.

1.2 Triaxial cyclic tests

1.2.1 Undrained triaxial with large strain cycle

Nine undrained triaxial cyclic tests have been performed with large strain amplitude. They are referred to as TCUEL. The range of ε_1 is $[5{\cdot}10^{-3}, 10^{-2}]$. Table 2.3 describes the protocols. All the tests are performed at a 0.05mm/min loading rate.

We have access to several quantities for TCUEL in the database, including ε_1, p, q, e_0.

1.2.2 Drained triaxial tests with large un- and reloading cycles

Seven drained triaxial cyclic tests have been performed. They are referred to as TMCD. The load is applied with a shearing rate of 0.1 mm/min. An unloading-reloading cycle is performed every $\Delta\varepsilon_1$. The initial density varies from 0.24 to 0.94, and p_0 from 50 kPa to 200 kPa. Table 2.4 describes the complete protocols.

We have access to several quantities for TMCD in the database, including $\varepsilon_1, \varepsilon_v, p, q, e_0$.

Test	e_0 (-)	I_{D0} (-)	p_0 (kPa)	$\Delta\varepsilon_1$ (%)
TMCD1	0.962	0.24	100	2
TMCD2	0.829	0.60	100	2
TMCD3	0.701	0.94	100	2
TMCD4	0.820	0.62	100	6
TMCD5	0.821	0.62	100	12
TMCD6	0.810	0.65	200	2
TMCD7	0.814	0.64	50	2

Table 2.3: Program of TCUE with large strain amplitude, [Wichtmann *et al.*, 2015].

Test	e_0 (-)	I_{D0} (-)	p_0 (kPa)	q_0/p_0 (-)	$\varepsilon_{\text{ampl}}$ ($\times 10^{-2}$) (10^{-2})
TCUE15	0.944	0.29	200	0	1.0
TCUE16	0.804	0.66	200	0	1.0
TCUE17	0.698	0.94	200	0	1.0
TCUE18	0.812	0.64	100	0	1.0
TCUE19	0.814	0.64	700	0	1.0
TCUE20	0.816	0.63	200	0	0.5
TCUE21	0.827	0.60	200	0.75	1.0
TCUE22	0.686	0.98	100	0	1.0
TCUE23	0.674	1.01	700	0	1.0

Table 2.4: Program of TMCD, [Wichtmann *et al.*, 2015].

1.3 Other types of tests

1.3.1 Drained isotropic compression tests

Six isotropic compression tests have been performed. They are referred to as ISO. The initial density varies from 0.21 to 0.99. Table 2.5 describes each loading path. Large cycles refer to the loading protocols where cycles vary between 50 kPa and 800 kPa. The first small cycle varies from 50 kPa to 800 kPa, but the unloading is smaller, with 200 kPa.

We have access to several quantities for ISO in the database, including ε_v, p, e_0.

Test	e_0 (-)	I_{D0} (-)	Cycles	p^{ampl} (kPa)
ISO1	0.974	0.21	large	750
ISO2	0.823	0.61	large	750
ISO3	0.690	0.97	large	750
ISO4	0.963	0.24	small	200
ISO5	0.824	0.61	small	200
ISO6	0.680	0.99	small	200

Table 2.5: Program of ISO, [Wichtmann *et al.*, 2015].

2 Data pre-processing

Herein, we describe the procedure implemented to preprocess the raw data from [Wichtmann *et al.*, 2015]. The database is composed of `.dat` files. Data are gathered over different protocols (see Sections 1.1.1, 1.2.2, 1.1.2, 1.2.1, 1.3.1). FIG.2.2 gives an overview of the preprocessing procedure. The first block, `import_to_pickle`, allows the transfer of data from `.csv` to binary files while filtering them. The second block, `NICE_XP`, is the new version of NICE that allows to learn from real data. It imports binary files and prepares them for use in NICE.

2.1 `import_to_pickle`

2.1.1 Import to Python

The parser first needs to convert `.dat` files to `.csv` to import the data to Python. After that, the parser reads `.csv` files and imports the data into a two-layer-deep dictionary, `raw_data_dict`. The keys are different quantities measured ($\varepsilon_1, \varepsilon_v, p, q, \ldots$), and the nested keys (the key associated with each quantity) denote the labeling of the loading paths.

2.1.2 Filtering

Data are filtered in order to reduce any eventual noise from the raw data. FIG.2.3 sums up the filtering process.

Given noisy input data, the aim is to remove the noise. Noise corresponds to high-frequency components in the Fourier spectra. To remove noise, we need to remove those high-frequency components. A low-pass filter is applied to the Fourier transform of the input noisy signal. The filter expression (2.3) depends on two main parameters – f_c, the cutoff frequency, and N, the order of the filter – and reads

$$X_f(f) = \frac{X(f)}{\sqrt{1 + \left(\frac{f}{f_c}\right)^{2N}}}. \tag{2.3}$$

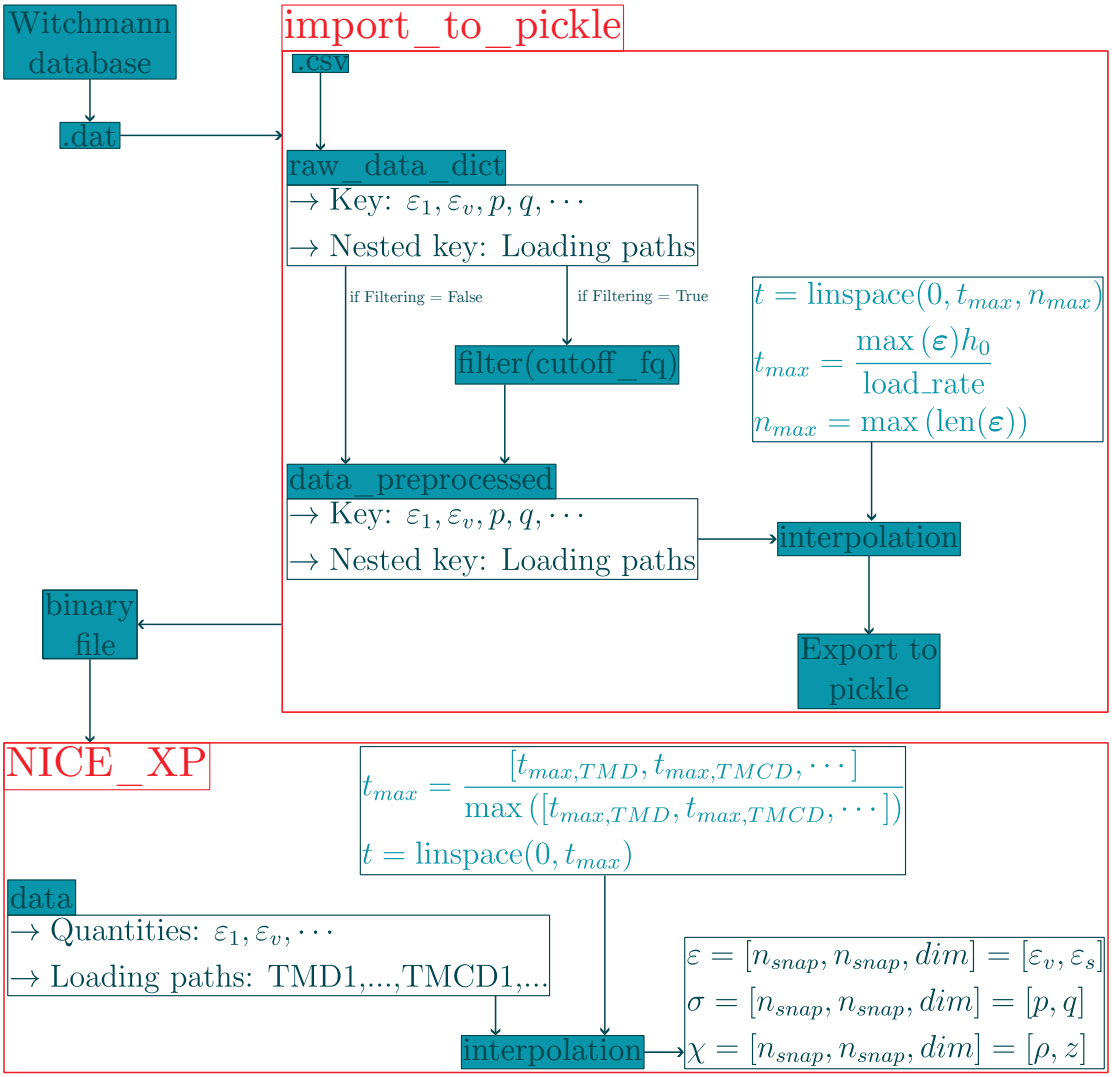

Figure 2.2: Overview of the preprocessing procedure.

FIG.2.3 shows an example of the effect of the filtering procedure.

2.1.3 Interpolation

In order to have the same number of snapshots for each loading path and allow for a straightforward implementation of the NICE framework, the data are interpolated with respect to the time and resampled the latter to the same time array. This time array is a linearly spaced point between 0 and t_{max}, where t_{max} is given by

$$t_{max} = \frac{\max(\varepsilon) h_0}{\text{load\_rate}}, \tag{2.4}$$

with h_0 the sample's height. The dataset is then stored in a binary file with the same number of snapshots for each loading path.

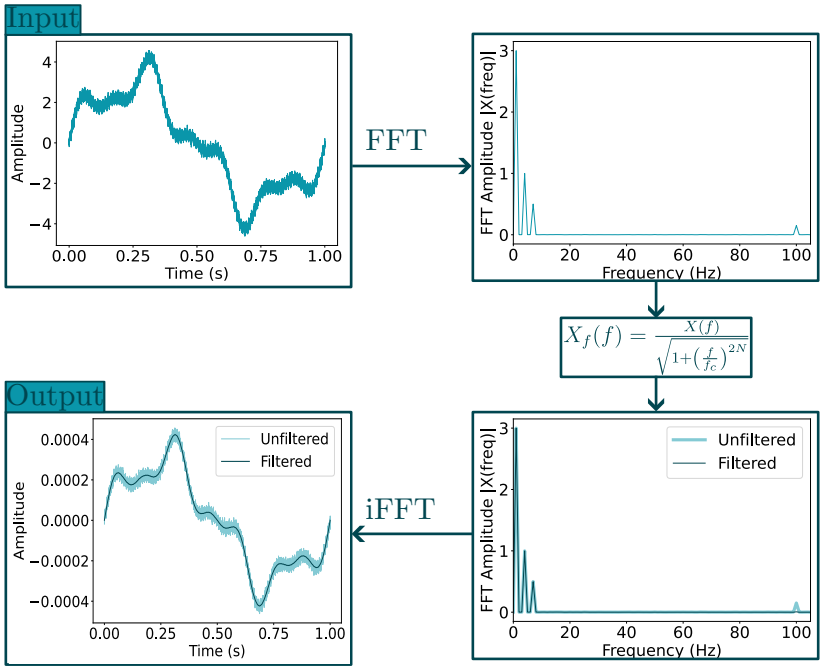

Figure 2.3: Overview of the filtering process.

2.2 NICE_XP

The data stored in the binary files are imported using different protocols (TMD, TMU, ISO, etc.). In order to ensure a consistent training of the machine learning algorithm, time arrays are scaled according to the procedure described in Fig.2.4.

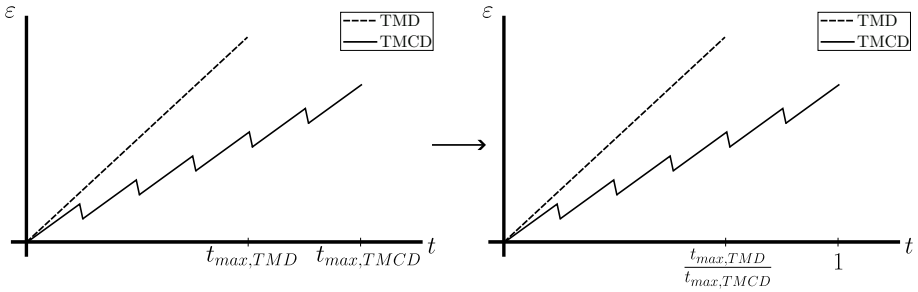

Figure 2.4: Time rescaling.

Fig.2.4 concept is implemented by interpolating the different datasets between 0 and $\frac{t_{max,\text{protocol}}}{\max(t_{max,\text{protocol}})}$. After that interpolation, datasets from the different protocols are concatenated together.

In reality, the NICE framework uses the strain rate in order to predict the evolution equations of the state variables. To keep using total strain as an input, we add a layer to compute the derivative from the data (see Algorithm 1).

Algorithm 1: Pseudocode for the computation of strain rates.

Require: $\varepsilon_v, \varepsilon_s, t$,path
for *proto in range(N_{proto})* **do**
    `interp_ex` = `BivariateSpline`(ε_x[proto], t[proto], path[proto]);
    `interp_dotex` = `interp_ex.partial_derivative`($dx = 1, dy = 0$)
return *interp_dotex*

3 Training

This section looks into the training phase and analyzes the results obtained.

3.1 TMCD + TMD

To expose the performances of NICE, we trained them on drained protocols, both TMD and TMCD. TMD contains twenty-five loading paths, and TMCD contains seven loading paths. According to Section 1.2.3, to have consistent results, we divide the dataset into training, validation, and test sets. 65% of the data are used for training, while the rest of the data are equally split between validation and test sets:

Protocol \ Phase	Training	Validation	Test
TMD	17	4	4
TMCD	5	1	1

Table 2.6: Number of paths per phase and per protocol.

3.1.1 Training parameters

For drained protocols, [Wichtmann *et al.*, 2015] database gives the values of both ε_1 and ε_3. However, this is not the case for undrained protocols. In the case of undrained triaxial, we assume that $\dot{\varepsilon}_v = 0$. Therefore, $\dot{\varepsilon}_s = \dot{\varepsilon}_1$ in the case of undrained. This assumption is made because we do not have data for ε_v from [Wichtmann *et al.*, 2015]. As presented at the beginning of this chapter, we only have access to the average material response, in terms of stress, strain, and volumetric density. To account for the internal structure, we additionally consider p dissipative (internal) state variable $z_1, \ldots, z_p$ (here, $p = 1$). The appeal of the NICE approach lies on the fact that only initial conditions for the material state are required. Accordingly, and without any loss of generality, we assume that the initial value of the additional internal variables is equal to zero, i.e., $z^{(0)} = 0$.
The training is done with twenty-two training paths and five validation paths. The architecture of NICE is respected, with two NN. The state variables to be described

are $\boldsymbol{\varepsilon}^e$, ρ, z. $\boldsymbol{f_\theta}$ is a three-layer FNN, with 'elu' as activation function. $\boldsymbol{\psi_\omega}$ is a two-layer FNN, with 'softplus' as the activation function. To enforce the convexity of the free-energy density, we apply the principles of Input Convex Neural Networks (ICNN) [Klein *et al.*, 2022]. The integration method used is 'dopri-5' [Dormand et Prince, 1980], which is an adaptive Runge-Kutta of order 5, integration method. The optimizer is ADAM [Kingma et Ba, 2014]. We also used an exponential scheduler to control the learning rate while we converged towards the solution.

3.1.2 Training

The training is conducted using an early stopping criterion based on the loss (1.31) associated with the validation data set to stop the training process and avoid overfitting. If after a given number of epochs, called *patience*, the validation loss did not decrease by more than δ, then the training stops (see Pseudocode 2).

Algorithm 2: Pseudocode for the training phase.

Require: network params $\{\boldsymbol{\theta}, \boldsymbol{\omega}\}$; hyper-params $\boldsymbol{\varepsilon}^{e(0)}$; optimizer.
Data: initial conditions $\boldsymbol{\chi}^{(0)} = \{\boldsymbol{\varepsilon}^{e(0)}, \rho, \mathbf{z}\}^{(0)}$; data and protocol $\{t^n, \boldsymbol{\sigma}^{(n)}, \boldsymbol{\varepsilon}^{(n)}\}_{n=1}^{N}$, $\{t^m, \mathbf{z}^{(m)}\}_{m=1}^{M}$.
while *training* **do**
  $\text{pred}_{\text{train}} = \text{NICE}(\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\chi}^{(0)}_{\text{train}}, \dot{\boldsymbol{\varepsilon}}_{\text{train}})$;
  $\mathcal{L}_{\text{train}} = \mathcal{L}(\text{pred}_{\text{train}}, \text{data}_{\text{train}})$;
  $\text{pred}_{\text{val}} = \text{NICE}(\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\chi}^{(0)}_{\text{val}}, \dot{\boldsymbol{\varepsilon}}_{\text{val}})$;
  $\mathcal{L}_{\text{val}} = \mathcal{L}(\text{pred}_{\text{val}}, \text{data}_{\text{val}})$;
  backpropagate $\mathcal{L}_{\text{train}}$;
  update $\{\boldsymbol{\theta}, \boldsymbol{\omega}, \boldsymbol{\chi}^{(0)}\}$;
  if $\mathcal{L}_{val} < \mathcal{L}_{val,min} - \delta$ **then**
    $\mathcal{L}_{\text{val,min}} = \mathcal{L}_{\text{val}}$;
    $\text{epoch}_{\text{min}} = \text{epoch}$;
  if *epoch* $>$ *epoch*$_{min}$ + *patience* **then**
    training false;
return *params* $\{\boldsymbol{\theta}, \boldsymbol{\omega}\}$

Fig.2.5 shows the value of the loss function, at training, for both the training and the validations set.

We can see that the training loss is lower than the validation loss. However, throughout the training phase, the validation loss keeps decreasing, indicating no overfitting.

In Fig.2.6, using a data science approach, we can plot the predicted value of deviatoric stress versus the true value of the deviatoric stress:

Fig.2.6 showcases that most of the points follow the reference line, indicating that the prediction is very accurate with respect to the true value. For both

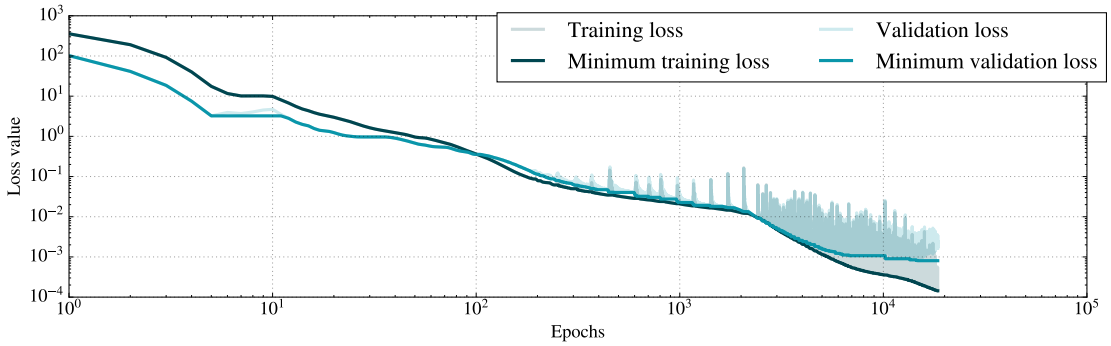

Figure 2.5: Loss value evolution for training and validation sets.

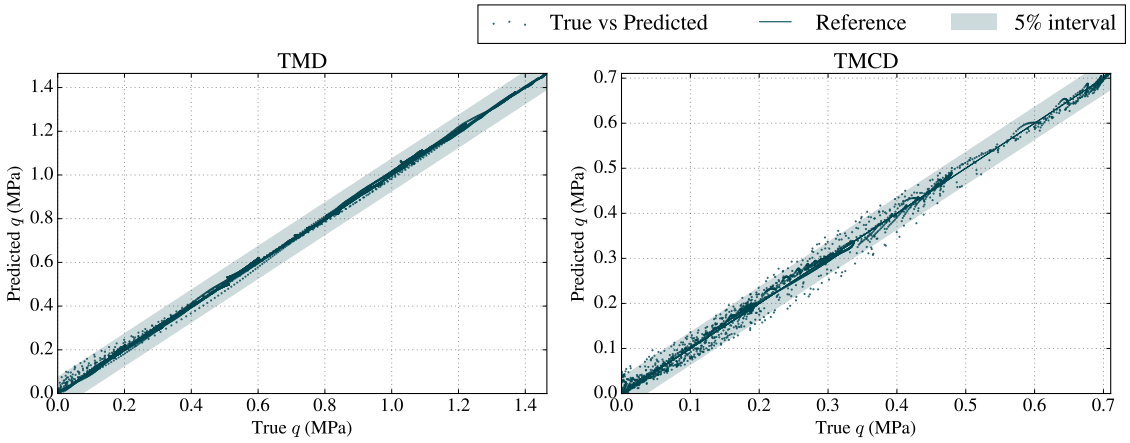

Figure 2.6: Scatter plot of the true deviatoric stress q versus the predicted deviatoric stress $q_{\boldsymbol{\theta},\boldsymbol{\omega}}$ at training.

protocols, all the points are within the 5% interval band.

FIG.2.7 showcases the deviatoric stress versus the deviatoric strain for various loading paths for the two protocols (monotonous and cyclic drained triaxial tests).

Plain lines refer to the predicted values, while shaded ones refer to true values. For TMD, we can see that the framework, at training, is capable of predicting the deviatoric stress at both different initial confining pressures and initial relative densities. Likewise, the cyclic predictions are excellent for different strain increments between cycles. We retrieve the results from FIG.2.6, where TMD is almost perfect, whereas TMCD has slight differences from the reference (for example in TMCD 6, we can see that at unloading the value of q is a bit overestimated). See Appendix A to have the complete training results. However, the results are outstanding; it is normal because the algorithm has access to both the input (being $\boldsymbol{\chi}^{(0)}$ and $\boldsymbol{\varepsilon}$) but also the output (being $\boldsymbol{\sigma}$).

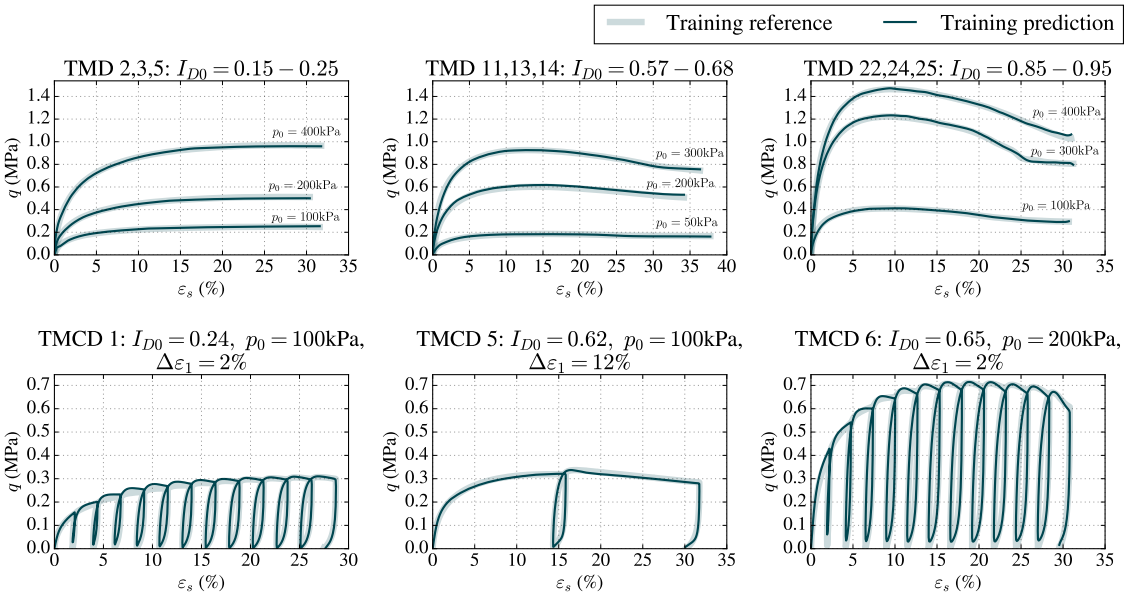

Figure 2.7: Prediction in the training phase, for both monotonous and cyclic drained tests, with different initial confining pressures and relative densities for monotonous drained tests, and different strain increments between cycles for cyclic drained tests.

3.1.3 Validation results

In this section, we explore the results obtained in the validation phase, which means that the algorithm cannot access the output. This phase is an important step that determines whether an algorithm can generalize to different data or not. We use the same plots to compare the results between training and validation.
Fig.2.8 illustrates, as excepted, that the results in the validation are not as good for both TMD and TMCD.In monotonous drained tests, almost all the points still follow the reference more or less the 5% interval. However, for cyclic, more points are outside that band.

Fig.2.9 confirms that result. For monotonous paths, we can see that the results are still good at different initial densities and (effective) volumetric stress. For the only cyclic path, we can see that the prediction is really good at a never-seen $\Delta\varepsilon_1$, even though the value of q is slightly underestimated. However, the overall quality of the predictions is excellent.

3.2 Analysis of internal state variables

In order to learn from TMD+TMCD, the state variables were augmented with an additional state variable, z, with zero initial value and arbitrary evolution identified during training. Fig.2.10 plots the evolution of z versus the deviatoric strain for two scenarios. The former consists of a monotonous drained triaxial path, and the

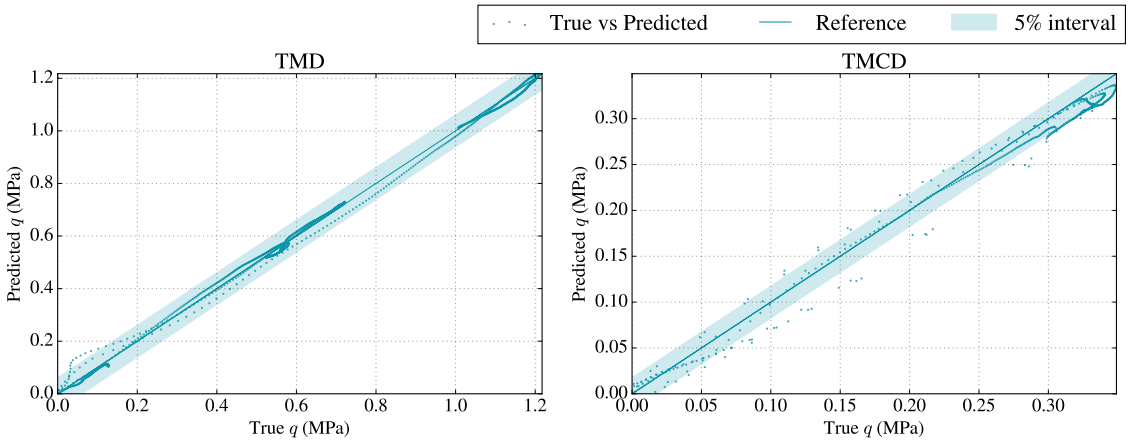

Figure 2.8: Scatter plot of the true deviatoric stress q versus the predicted deviatoric stress $q_{\theta,\omega}$ at validation.

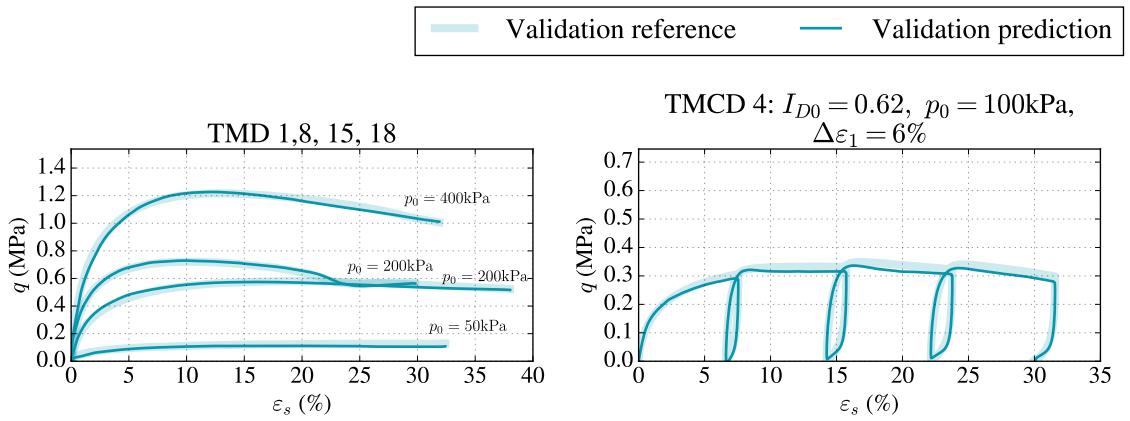

Figure 2.9: Prediction in the validation phase, for both monotonous and cyclic drained tests, with different initial confining pressures and relative densities for monotonous drained tests, and a strain increment of $\Delta\varepsilon_1 = 2\%$ between cycles for the cyclic drained test.

latter is a cyclic triaxial with the same initial volumetric stress and similar relative densities. Note that the evolution of the internal state variable is very similar for the two cases upon the first unloading/reloading cycle (at $\varepsilon_s \approx 2$ %). Afterward, z evolves differently depending on whether multiple cycles are performed. It is particularly interesting to notice that, throughout the multiple unloading/reloading cycles, the internal state variable seems evolving towards a new (meta-)stable state characterized by a larger value compared to the simple monotonous evolution. This seems to suggest that the additional variable describes the instantaneous change of stiffness observed during cyclic loading, where the area comprised by each closed loop of the evolution of z highly correlates with the area beneath the stress-strain

curve during unloading/reloading. Nevertheless, in the absence of any microscopic information related to the tested material, it remains extremely difficult to assess and characterize the physical nature (if any) of such additional variables, and more advanced experimental tests should be performed to achieve physical descriptions of the material state space (cf. Chapter 3).

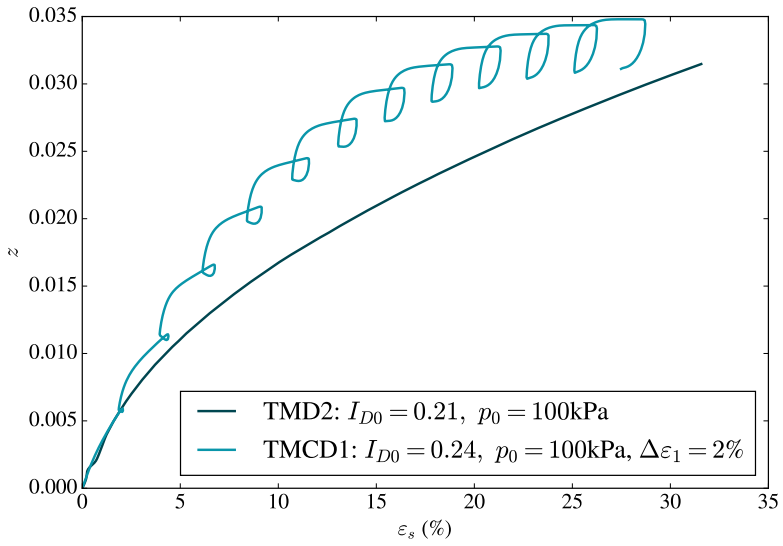

Figure 2.10: Evolution of the additional internal state variable, z, for monotonous and cyclic, with similar initial conditions.

3.2.1 Tests

Test set

This first test uses a part of the dataset TMD-TMCD, called the test set. It is similar to the validation set. The test set is usually used after the training to assess the quality of the neural network. Fig.2.11 showcases the performances of the network in predicting unobserved loading paths. However, note that some predictions lie outside of the 5% band for the cyclic test.

Fig.2.12 showcases the same behavior. The quality of TMD is really good at various initial densities and pressures. However, the value of q for the TMCD path is underestimated at high deviatoric strain. It might be due to an accumulation of integration errors or a lack of capability of the network.

3.3 All

Now that we have been able to learn from experimental data for similar protocols, what would happen if we tried to learn from various protocols. In this section, we try to learn from TMD, TMCD, TMU, TCUEL, and ISO (all the protocols presented

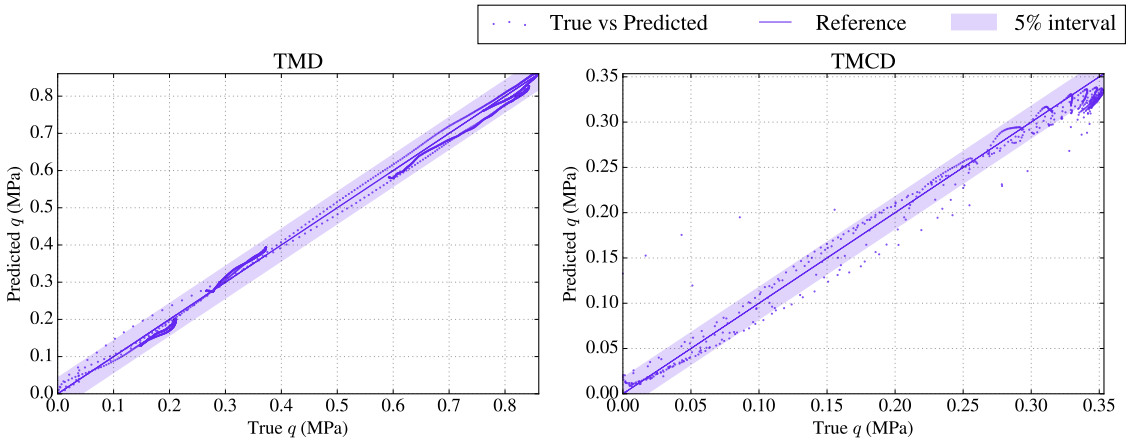

Figure 2.11: Scatter plot of the true deviatoric stress q versus the predicted deviatoric stress $q_{\theta,\omega}$ at testing.

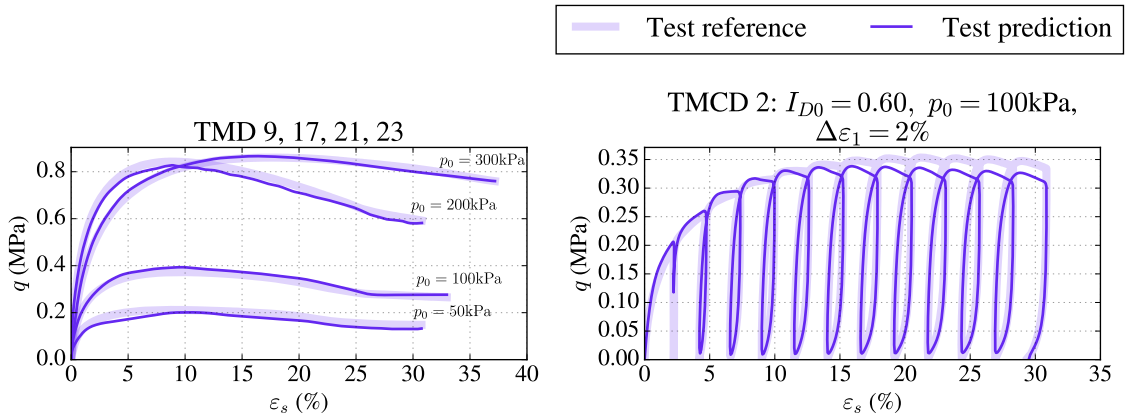

Figure 2.12: Prediction in the test phase, for both monotonous and cyclic drained tests, with different initial confining pressures and relative densities for monotonous drained tests, and a strain increment of $\Delta\varepsilon_1 = 2\%$ between cycles for the cyclic drained test.

in 1). The architecture used for NICE is the same as the one used in Section 3.1. This first FIG.2.13 illustrates the evolution of the loss during the training phase.

First results can be extracted from FIG.2.13. It is really clear that there is overfitting. Training loss keeps decreasing while validation loss increases. We, therefore, expect fairly good training results but really poor validation results.

3.3.1 Training results

FIG.2.14 showcases the results obtained after the training phase for the training set.

The network learns most of the protocols. We can retrieve the loading paths for

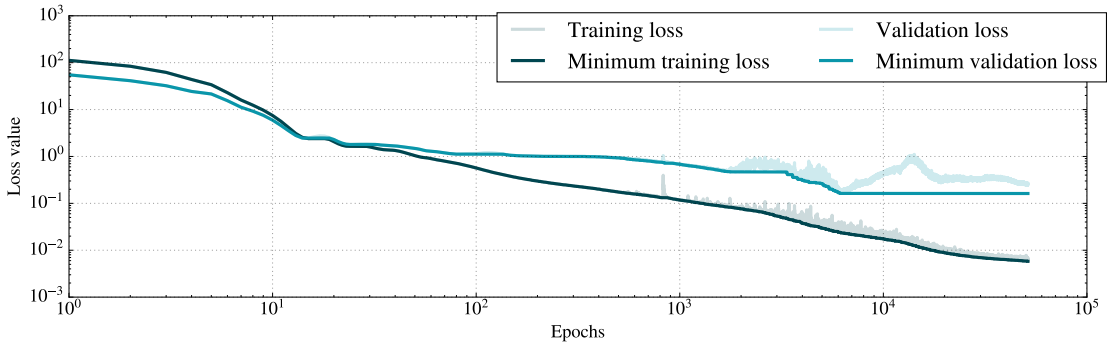

Figure 2.13: Loss value evolution for training and validation sets.

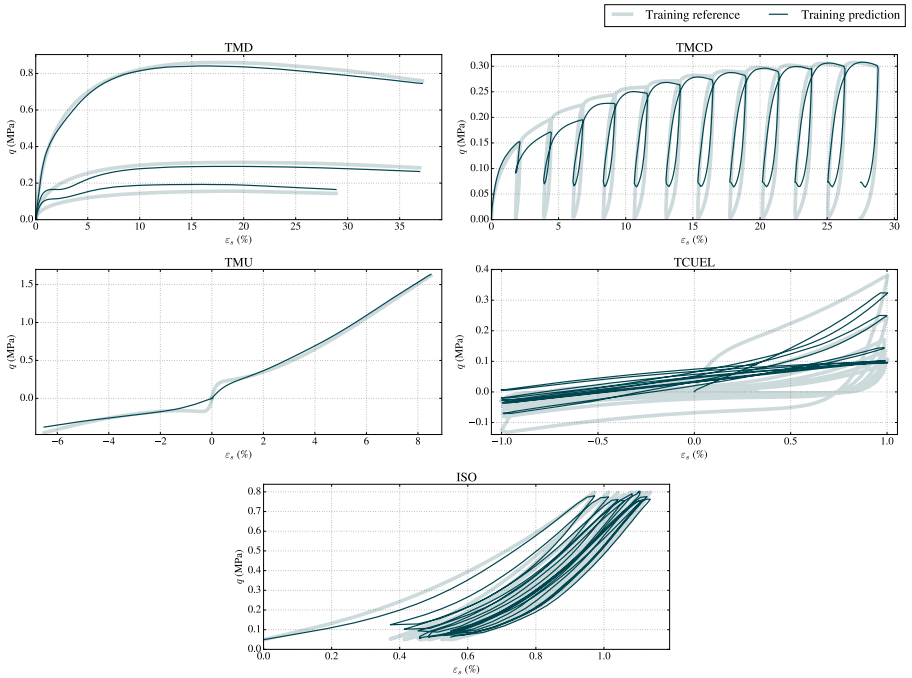

Figure 2.14: Prediction in the training phase, for all the protocols, with various initial conditions.

TMD, TMCD, TMU, and ISO. The framework does not understand TCUEL at all. Overall, the framework could be improved to obtain better results by modifying the framework itself or extending the training phase.

3.3.2 Validation results

FIG.2.15 showcases the results obtained after the training phase for the validation set.

This figure confirms that the network largely overfits the training data. The qual-

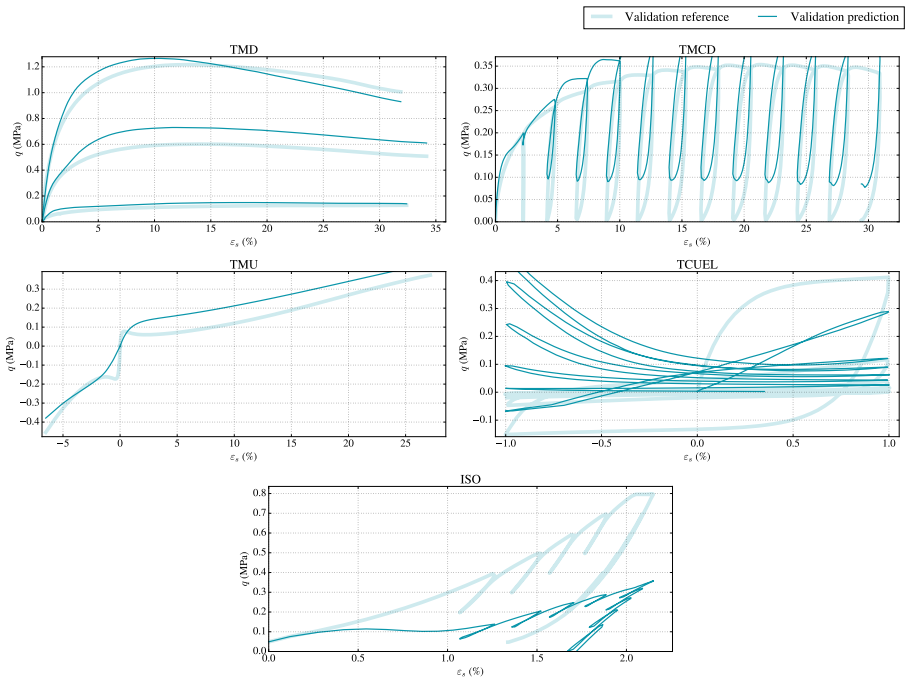

Figure 2.15: Prediction in the training phase, for all the protocols, with various initial conditions.

ity is poor compared to those obtained for the training set and those obtained when only training on drained tests. For example, we can see that the results obtained for ISO do not make any sense. The values for TMCD are also overestimated.

Chapter 3

Conclusions and perspectives

The objective of this research project was to investigate the possibility of discovering material constitutive equations directly from physical experiments of a complex, heterogeneous material, i.e., sand, via physics-based, data-driven approaches and, in particular, the Neural Integration for Constitutive Equations (NICE) approach [Masi et Einav, 2024].

1 Key findings

Chapter 1 presented the NICE methodology, including an overview of how machine learning and neural networks work. Here, additional constraints derived from physics and mechanics were introduced, namely the convexity (by construction) of the free energy function with respect to the state variables which results in the well-posedness of the underlying boundary value problem (computational mechanics).

Chapter 2 focused on the main results of this work. First, the considered experiments and the associated database [Wichtmann *et al.*, 2015] were presented and summarized. Second, the pre-processing pipeline for handling the raw data was presented and its performance discussed. Then, the revised NICE framework was applied for learning constitutive equations from conventional drained triaxial (monotonous and cyclic) tests.

The NICE approach has demonstrated that it is possible to discover constitutive equations from conventional experimental tests on an emblematic geomaterial (dry sand), without the need for finely sampled acquisitions of the material state or probing of microstructural quantities (displacement/velocity fields, contact fabric, etc.). The approach was validated using unobserved loading paths (validation sets). It was found able to extrapolate to unseen data for drained triaxial tests showing remarkable and promising results. Nevertheless, the current framework was not able to accurately learn the material response under isotropic compression and undrained triaxial experiments, due to the amount/quality of data and the lack of information pertaining to the internal microstructure of the material (besides total strain, stress, and volumetric mass density).

2 Perspectives

Further investigations and developments of the presented work can be pursued. The following steps are supposed to enable the data-driven discovery of constitutive equations from experimental observations:

1. Have access to extensive experimental campaigns carrying numerous experiments with varying loading paths, protocols, and initial conditions. Whilst the database herein considered [Wichtmann *et al.*, 2015] is one of the most comprehensive in the field of geomechanics, a relatively limited set of initial confining pressures and densities is available. This may hinder the capabilities of data-driven approaches due to the fact that large, non-negligible regions of the material state space (state variables) are not probed. In such configurations, data-centric approaches are thus required to extrapolate not only the material stress-strain response but also the entire state space (density, elastic strain, etc.) and its time evolution. More complete databases could thus reduce the issues associated with such an extrapolation task.

2. Have access to microstructural material information and not only average descriptors. In the presented work, we leveraged additional internal (dissipative) state variables to enable the modeling of the material response under cyclic drained triaxial tests (e.g. hysteresis). However, no information was available to define a proper physical interpretation for such internal variables. Performing experiments with advanced three-dimensional imaging techniques (e.g., X-ray tomography/diffraction [Hall *et al.*, 2011, Andò *et al.*, 2022]) would enable to track not only external forces and displacements but also the microstructure and its evolution (e.g., grain displacements/rotations, contact fabric). Such detailed representations could then be used either to compute phenomenological state variables or identify latent (state) variables on which the constitutive equations can be formulated, cf. [Masi et Stefanou, 2022, Masi, 2023b].

3. Hard-wire additional constraints and principles stemming from mechanics, physics, and thermodynamics. In the present work, we only considered the Clausius-Duhem inequality (energy balance and entropy production), the mass balance, and the convexity of the free-energy density. Leveraging richer physical principles and (general) theories from solid mechanics, such as hydrodynamics [Einav et Liu, 2018], will undoubtedly enable more accurate descriptions of the material response and increased capabilities at extrapolation.

Addressing and leveraging the aforementioned steps is expected to enable more robust and generalizable ML models capable of effectively learning complex material behaviors. This is currently investigated in an ongoing work and a manuscript (cf. Curriculum Vitae) in collaboration with Prof. Itai Einav and Dr. Filippo Masi where we enforce additional constraints derived from physics to considerably improve the predictions and results that I obtained and presented within this Thesis.

Bibliography

[Andò *et al.*, 2022] Andò, E., Marks, B., Hurley, R. et Dijksman, J. A., éditeurs (2022). *ALERT Doctoral School 2022: Advanced Experimental Geomechanics*.

[Boyd et Vandenberghe, 2004] Boyd, S. et Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[Dormand et Prince, 1980] Dormand, J. et Prince, P. (1980). A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

[Einav et Liu, 2018] Einav, I. et Liu, M. (2018). Hydrodynamic derivation of the work input to fully and partially saturated soils. *Journal of the Mechanics and Physics of Solids*, 110:205–217.

[Gatti, 2023] Gatti, F. (2023). Artificial neural networks: layer architectures, optimizers and automatic differentiation. *In* [Stefanou et Darve, 2023], pages 159–253.

[Ghaboussi *et al.*, 1991] Ghaboussi, J., Garrett, J. H. et Wu, X. (1991). Knowledge-based modeling of material behavior with neural networks. *J Eng Mech*, 117(1):132–153.

[Goodfellow *et al.*, 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. et Bengio, Y. (2014). Generative adversarial networks.

[Gu *et al.*, 2020] Gu, X., Huang, M., Qian, J. et Zhao, C. (2020). An ai-based model for describing cyclic characteristics of granular materials. *International Journal for Numerical and Analytical Methods in Geomechanics*, 44(6):860–878.

[Hall *et al.*, 2011] Hall, S. A., Wright, J., Pirling, T., Andò, E., Hughes, D. J. et Viggiani, G. (2011). Can intergranular force transmission be identified in sand? first results of spatially-resolved neutron and x-ray diffraction. *Granular Matter*, 13:251–254.

[Ideami, 2019] Ideami, J. (2019). Loss Landscape A.I Project. https://losslandscape.com/explorer.

[Joseph, 2022] Joseph, V. R. (2022). Optimal ratio for data splitting. *arXiv preprint arXiv:2202.03326*.

[Karniadakis *et al.*, 2021] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S. et Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440.

[Kidger, 2022] Kidger, P. (2022). On neural differential equations.

[Kingma et Ba, 2014] Kingma, D. P. et Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Klein *et al.*, 2022] Klein, D. K., Fernández, M., Martin, R. J., Neff, P. et Weeger, O. (2022). Polyconvex anisotropic hyperelasticity with neural networks. *Journal of the Mechanics and Physics of Solids*, 159:104703.

[Masi, 2023a] Masi, F. (2023a). Introduction to regression methods. *In* [Stefanou et Darve, 2023], pages 29–75.

[Masi, 2023b] Masi, F. (2023b). Physics-informed and thermodynamics-based neural networks. *In* [Stefanou et Darve, 2023], pages 351–383.

[Masi et Einav, 2024] Masi, F. et Einav, I. (2024). Neural integration for constitutive equations using small data. *Computer Methods in Applied Mechanics and Engineering*, 420:116698.

[Masi et Stefanou, 2022] Masi, F. et Stefanou, I. (2022). Multiscale modeling of inelastic materials with Thermodynamics-based Artificial Neural Networks (TANN). *Computer Methods in Applied Mechanics and Engineering*, 398:115190.

[Masi et Stefanou, 2023] Masi, F. et Stefanou, I. (2023). Evolution tann and the identification of internal variables and evolution equations in solid mechanics. *Journal of the Mechanics and Physics of Solids*, 174:105245.

[McCulloch et Pitts, 1943] McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill series in Computer Science. McGraw-Hill, New York, nachdr. édition.

[Poblete *et al.*, 2016] Poblete, M., Fuentes, W. et Triantafyllidis, T. (2016). On the simulation of multidimensional cyclic loading with intergranular strain. *Acta Geotechnica*, 11.

[Prechelt, 1998] Prechelt, L. (1998). Early stopping - but when? pages 55–69.

[Radford *et al.*, 2018] Radford, A., Narasimhan, K., Salimans, T. et Sutskever, I. (2018). Improving language understanding by generative pre-training. *OpenAI*.

[Rosenblatt, 1957] Rosenblatt, F. (1957). The perceptron - a perceiving and recognizing automaton. (85-460-1).

[Schmidhuber, 2014] Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117.

[Sebastian Raschka, 2022] Sebastian Raschka, Yuxi (Hayden) Liu, V. M. (2022). *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing.

[Silver *et al.*, 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. *et al.* (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[Solomonoff, 1957] Solomonoff, R. J. (1957). An inductive inference machine. *IRE Convention Record, Section on Information Theory, Part 2*, pages 56–62.

[Stefanou, 2023] Stefanou, I. (2023). Overview of machine learning. *In* [Stefanou et Darve, 2023], pages 5–29.

[Stefanou et Darve, 2023] Stefanou, I. et Darve, F., éditeurs (2023). *ALERT Doctoral School 2023: Machine Learning (ML) in Geomechanics*.

[Tromp et Farnebäck, 2007] Tromp, J. et Farnebäck, G. (2007). Combinatorics of go. 4630:84–99.

[Wichtmann *et al.*, 2015] Wichtmann, T., Niemunis, A. et Triantafyllidis, T. (2015). An experimental database for the development, calibration and verification of constitutive models for sand with focus to cyclic loading. *Acta Geotechnica*, 10(5):545–563.

Appendix A

Drained triaxial predictions

This section showcases all the results with the different drained protocols and training paths.

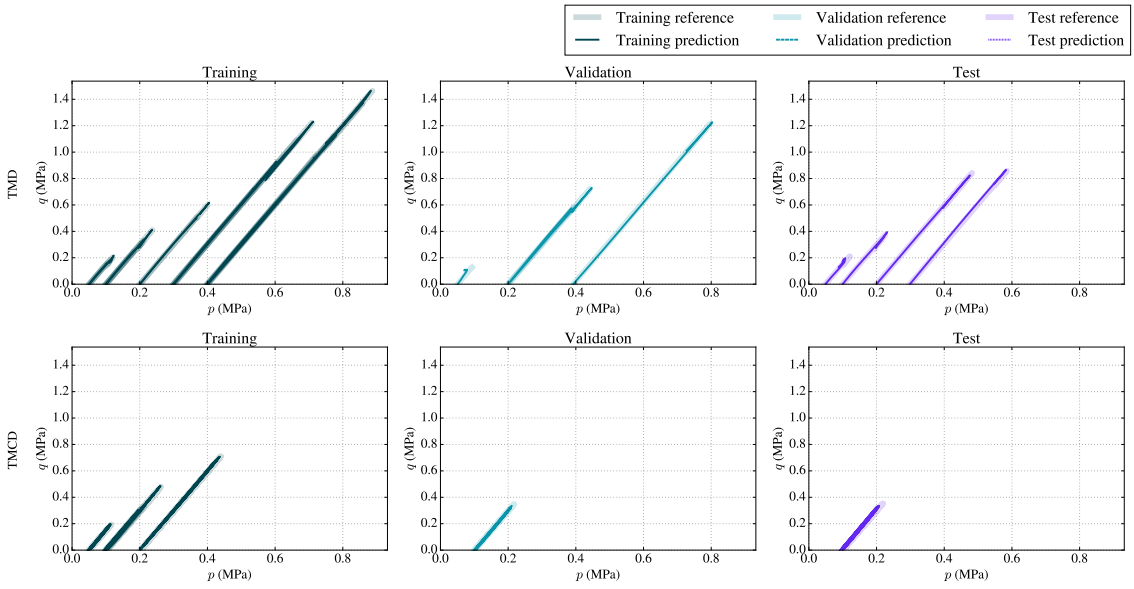

Figure A.1: Effective stress paths in the p-q-plane, measured and predicted, for drained triaxial tests with different initial pressures and densities.

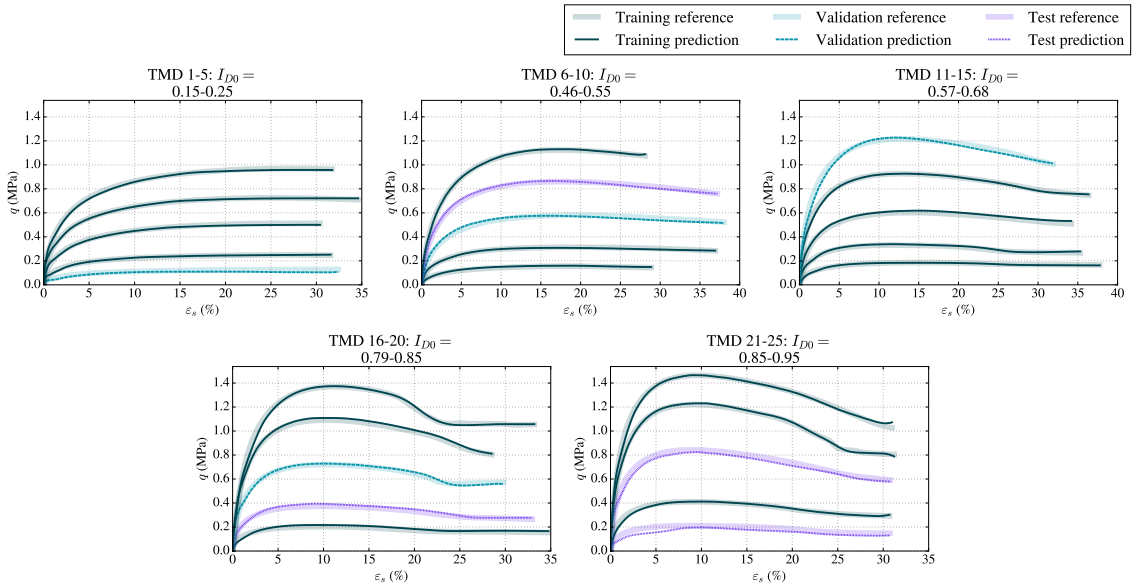

Figure A.2: Prediction for monotonous drained tests, with different initial confining pressures and relative densities

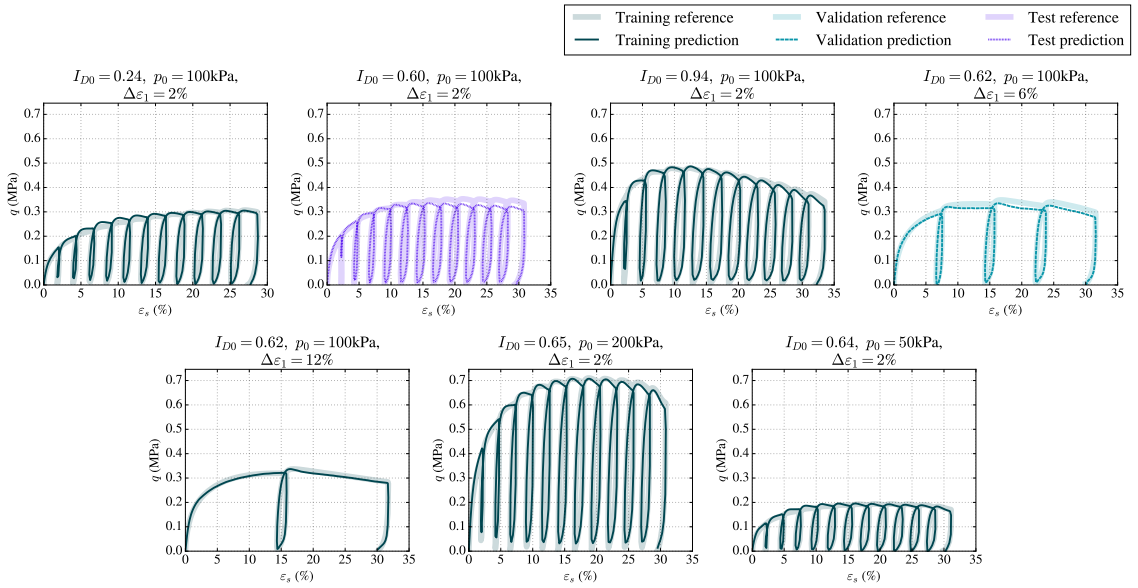

Figure A.3: Prediction for cyclic drained tests, with different initial confining pressures, relative densities and strain increments between cycles

ENZO LOUVARD

A French student of Civil Engineering and Artificial Intelligence

enzo.louvard@ens-paris-saclay.fr +33 6 17 70 03 81 ENS Paris Saclay - ENSTA, France Enzo Louvard enzolvd
38 rue de Tours, 37270 Saint Martin le Beau, France

EDUCATION

Currently in third year in GC, Génie Civil, at the Master's 2 GCER, Geomechanics Civil Engineering and Risks

Phitem, Université Grenoble Alpes

Date: september 2023 – now Location: Grenoble, France

- The third year is a second year of master's decicated to research, called GCER, Geomechanics Civil Engineering and Risks. Courseworks are Continuum Mechanics, Numerical Methods for non linear mechanics, Experimental Methods in Geomechanics, Basic Geomechanics, Basic Engieering Seismology, Quantitative Image Analysis for Mechanics and Soil Dynamics and Non Linear Site Response Analysis. Equivalent to a Master's Degree in structural Civil Engineering. 60 ECTS

ENS Paris-Saclay, Ecole normale supérieure de Paris-Saclay, Université Paris-Saclay

Date: september 2022 – august 2023 Location: Gif-sur-Yvette, France

- The second year is a master of science in civil engineering, called MAISES, Matériaux et Structure (Materials and Structure). Courseworks are Continuum Mechanics, Structure Calculation, Building Methods, Vibrations of Solids and Seismic Engineering, Soil Mechanics, Numerical Methods, Machine Learning, Development of Simulation Tools, and Modelization of Quasi-brittle Materials. Equivalent to a Master's Degree in structural Civil Engineering. 60 ECTS

ENS Paris-Saclay, Ecole normale supérieure de Paris-Saclay, Université Paris-Saclay

Date: september 2021 – June 2022 Location: Gif-sur-Yvette, France

- A year in general engineering science called SAPHIRE. Courseworks are Mathematics, Mechanics, Electromagnetism, Material studying, Computer science and Signal processing. Equivalent to a Bachelor's Degree in Engineering Science. 60 ECTS.

Classe Préparatoire MPSI-PSI*, Lycée Descartes

Date: september 2019 – june 2021 Location: Tours, France

- Intensive two-year study preparing for the competitive entrance examinations to the French "Grandes Écoles" (the top French and highly-selective institutions), 120 ECTS.
- Courseworks are Physics, Mathematics, Engineering Science and Computer Science.

Obtained the *Baccalauréat* in Science, specialty mathematics summa cum laude

Lycée Léonard de Vinci

Date: september 2016 – july 2019 Location: Amboise, France

WORK EXPERIENCE

Internship: Physics-based, data-driven discovery of constitutive equations for granular media, Supervisors: Filippo Masi, Itai Einav

Sydney Centre in Geomechanics and Mining Materials, The University of Sydney

Date: february 2024 – july 2024 Location: Sydney, Australia

- Collect and preprocess data from repositories,
- Development of new knowledge in deep learning algorithms for the constitutive modeling of materials,
- Advanced coding and mathematical development in neural differential equations and thermodynamics of continua.
- Research Paper to be published with Prof. Einav and Dr. Masi
- Poster session at SANDLESS 2024, Q-Station, Sydney

Internship: Anomaly detection in time series using bayesian machine learning methods, Supervisor: James.-A Goulet

Département des génies civil, géologique et des mines (CGM), Polytechnique Montréal

Date: may 2023 – july 2023 Location: Montréal, Canada

- Use of machine learning methods applied to the monitoring of structures, such as dams and bridges, on data from these types of structures
- Reading of theoretical literature and reproduction of results already obtained in order to become familiar with the machine learning methods used,
- Implementation of new methods and comparison of their performance with existing methods,

PROJECT EXPERIENCE

Image correlation applied to dynamic of structures

ENS Paris-Saclay, Ecole normale supérieur de Paris-Saclay, Université Paris-Saclay

Date: october 2022 – april 2023 Location: Gif-sur-Yvette, France

During my PEIR (Première Experience Immersive de Recherche) I had the opportunity to take a first step in the academic research.

- Research project in structures' dynamics and image correlation in cooperation with a senior R&D researcher at EDF (Mr. G-Hervé-Secourgeon)
- Aimed to develop a simple indicator of damage on structures that had been stressed in seismic bending, using image correlation. It uses an equivalent linear approach to determine quantities of interest (Young's modulus, damping coefficient).

Automation of a green house

ENS Paris-Saclay, Ecole normale supérieur de Paris-Saclay, Université Paris-Saclay

Date: october 2021 – june 2022 Location: Gif-sur-Yvette

Created and controlled a greenhouse automatically to produce organic tomatoes by optimizing the growing conditions for a maximum yield.

- Design of a PCB
- Wired an electric rack
- Designed an opening window with a jack

Numerical method applied to a thermal problem

Lycée Descartes

Date: october 2020 – april 2021 Location: Tours, France

In the context of my TIPE (Travaux d'Initiative Personelle Encadré), I made a project concerning conduction in solids.

- Implemented an algorithm based on a finite differences method to solve a thermal conduction problem
- Designed an experimentation to compare the result of the numerical resolution

SOFTWARE

Office Suite — Level :
Very Proficient

Python Library pyTorch — Level :
Highly Proficient

LANGUAGES

French — Level : Mother tongue

English — Level : C1
Work Proficiency

PROGRAMMING LANGUAGES

LateX — Level :
Highly Proficient

Python — Level :
Highly Proficient

Matlab — Level :
Very Proficient

C++ — Level :
Proficient

HOBBIES

- **Skiing and snowboarding**: Practised skiing every year since the age of 7 and snowboarding since the age of 14.
- **Track and field and triathlon**: Practising for 6 years in competition
- **Rugby**: Played at a university level for ENS Paris-Saclay and UGA
- **Handball**: Played for 10 years in competition

PERSONAL SKILLS

- **Computer skills**: Excellent Command of Matlab, a programming and simulation software, and good command of Office Suite (Word, Excel, PowerPoint). Excellent knowledge of Python. Very proficient in LateX, a programming language to write and layout all kinds of scientific papers.
- **Machine Learning skills**: Skilled to process data: reading, analysis, conversion, graphic representation, image processing. Capabilities to build advanced, multi-agent networks. Training focused on choosing the right machine learning tool according to the problem being studied. Also trained to evaluate the relevance of learning: cross-validation, overfitting, local and global minimums,...
- **Practical Works** Set up an experiment, follow a experimental protocol, discuss hypothesis used, write a report on a practical course
- **Other Civil Engineering Skills**: Capable of dimensioning and building a reinforced concrete beam. Large knowledge of classical tests to determine characteristics of a material.
- **Driver's licence** Holder of a driving licence since 2019.

REFERENCE

Available on request