Università degli studi di Milano-Bicocca

DEPARTMENT OF MATERIALS SCIENCE
Doctorate in Materials Science and Nanotechnology - XXXVI Cycle

Deep Learning Methods for the Investigation of Temporal Evolution of materials

Cadidate:
Daniele Lanzoni

Thesis advisor:
Prof. Francesco Montalenti

Vice-Coordinator:
Prof. Cristiana Di Valentin

Academic year 2022/2023

CONTENTS

INTRODUCTION

This Thesis has been completely written by a human.

In the last years, we have witnessed a true "Machine Learning revolution". Under the name of Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL), statistical methods have shown that many complex tasks can be solved automatically if data are abundant and are used in a clever way. In many cases, such as Natural Language Processing, Computer Vision, and recommendation systems, Machine Learning is already a standard tool. In other cases, applications are still in their infancy, despite already showing impressive results. It is almost impossible to make a list of such breakthroughs.

Computational materials science has not been immune to the Artificial Intelligence fever [1, 2]. More and more research groups are now trying to integrate some degree of Machine Learning algorithms in their work. While some applications are very close to the purely data-driven context ML methods were originally developed, such as non-linear fitting [3, 4], image elaboration [5], composition and properties search [6, 7, 8, 9], other approaches search for deeper integration with the underlying physical processes or mathematical models [10, 11, 12, 13, 14]. In many instances, ML algorithms can provide flexible, convenient and accurate function approximators. In this sense, they promise to beat traditional schemes in terms of scale: maybe the most successful example is that of so-called Deep Learning potentials, in which one tries to substitute expensive Density Functional Theory calculations with cheaper Neural Network approximations to perform high-accuracy molecular dynamics simulations at a reduced computational cost [15, 16, 17, 18]. Another hope is the possibility of integrating theoretical results and experimental data: Machine Learning algorithms are agnostic on the origin of the data. It is at least in principle possible to train models with hybrid computational/experimental datasets. In such a framework, ML can be a shortcut to finding new patterns, creating surrogate models, finding correction terms for known theories, etc. All of these can be attempted, however, only if we understand how to merge statistical learning with computational materials science, condensed matter physics and statistical physics.

In this Thesis, I will present my effort, focused on the application of Neural Networks methods to materials simulations. In particular, I leveraged the historical experience of my research group in multi-scale modeling. The main objective of my PhD was to try to integrate ML methods in continuum scale simulations involving mesoscale behaviors such as dislocations or morphological evolutions. The scope is methodological: my objective for the past three years has been to find new ways to tackle problems with known

bottlenecks in computational materials science. In all topics discussed, NNs have been implemented from scratch using PyTorch framework [19].

Machine Learning, especially if considered applied to this class of problems, is a relatively novel field. For this reason, specific textbooks on the applications of DL to computational physics/materials science are scarce. An attempt has thus been made to provide a self-contained discussion, presenting at least an overview of all theoretical concepts while avoiding exceeding in too-technical details. Of course, basic tools for a physicist are assumed to be familiar (differential equation, functional analysis, probability theory, etc). At the same time, as NN models leverage a lot of the mathematical formulation of physical models, at least general details of those are also revised. In the end, the Thesis contains a good amount of material. For this reason, we suggest the reader familiar with specific topics to skip introductions and go directly to results.

I find it convenient to introduce Deep Learning methods contextually to their applications. The hope is that by reading through the text, the reader can appreciate how ML models emerge naturally from the characteristics of the problems at hand. Of course, Chapter 1 represents an exception to this rule: before diving into applications, a short overview of basic concepts in Machine Learning and Neural Networks will be provided, together with the essential mathematical instruments. After this introduction, I will not proceed in strict chronological order in what I have done during my PhD. Instead, methods and applications will be introduced progressively from the most simple to the most sophisticated.

Chapter 2 will show an application of the staple of NN architectures, the multilayer perceptron, to approximate interaction energy and forces between dislocations in a semiconductor film. This idea of approximating a driving force for a system evolution with NNs is extended to the elastic energy driving the morphological evolution of strained films in chapter 3. As this quantity has an intrinsic spatial structure, this is a natural setting for introducing Convolutional Neural Networks.

The following Chapter 4 is the most dense one, as it moves to more sophisticated ML models: the idea is that it is possible to use a Neural Network to provide the dynamics predicted by a suitable Partial Differential Equation (PDE) without explicitly predicting the driving force or time derivatives. In this context, subsequent states of the system are regarded as elements of a sequence and a Recurrent structure is composed with the Convolutional NN to provide a fast approximation of traditional solvers. This scheme is presented in the context of Phase Field models. Additionally, a simple but effective scheme describing how prediction uncertainty estimations can be obtained in ML schemes is discussed. The Chapter also presents some preliminary results which are currently being collected for publication.

The final part of the Thesis is dedicated to the possibility of employing NN to learn statistical distributions. In particular, during my PhD, I focused on the possibility of using Generative Adversarial Networks (GANs) to directly generate stochastic processes described by a Markov chain. Chapter 5 sets the foundation for this scheme for the toy problem of a single particle diffusing in a double well potential and then outlines the work in progress in extending results to many particles, two-dimensional systems.

ACKNOWLEDGMENTS

Results in this work would not have been possible without the help of many people I collaborated with during my PhD. Machine Learning is a discipline that thrives when data are abundant, but data must come from somewhere. These people provided essential tools, setting the foundations on top of which this Thesis could be built. I am very indebted to all of them.

First, my supervisor, Prof. Francesco Montalenti provided the perfect environment in which my interests could develop. He took the bet of moving from traditional, well-established applications to jump into this new field, accompanying and trusting me on new, crazy proposals. Most of the considerations in this Thesis come from the countless hours of discussion with him in front of a blackboard.

During my period abroad in Lyon, I collaborated with Prof. Olivier-Pierre Louis. I learned a lot from him about statistical physics and stochastic systems. The whole work of Chapter 5 would only be a suggestive idea without his precious inputs, remarks, and attention to detail.

I wasted many Fabrizio Rovaris' afternoons with questions on elasticity and dislocations. He provided and adapted the group FEM codes for generating training sets. Without him, in all this Thesis there would not be a single elastic numerical calculation.

Roberto Bergamaschini was the person I referred to for all technical questions about Phase Field simulations and sharp interface models. He followed me into the rabbit hole of many numerical implementations and provided the KMC code for the many interacting particle system in Chapter 5.

Marco Albani ran simulations for the high-quality Phase Field surface diffusion dataset in Chapter 4. Luis Martìn-Encinar and Andrea Fantasia provided important help in running simulations and generating the dataset for Chapter 3 and the spinodal decomposition one of Chapter 4 respectively.

1 MACHINE LEARNING FUNDAMENTALS

In this Chapter, we will provide a quick introduction to the fundamentals of Machine Learning and Neural Networks. First, a brief description of what we mean by Machine Learning, what are its objective and some basic terminology is provided (Section 1.1). In Section 1.2, we will dive a bit deeper into the mathematics behind Machine Learning. The main actor here will be the concept of *loss function*. This quantity has deep connections with probability, but we will start slowly and first set things in a very heuristic framework.

A discussion on probabilistic interpretations will instead be tackled in Section 1.3. It is assumed that the reader is familiar with basic probability and statistics concepts, such as expectation, variance, probability distributions and densities, absolutely continuous variables, etc. A refresher can be found in [20]. Key results and theorems will however re-stated in the notation used in this Thesis.

Having set the game, the simplest example of Machine Learning methods, Linear Regression, is reviewed in Section 1.4 and Section 1.5. The first one defines and outlines the main characteristics of linear models. In doing this, we will be mainly interested in outlining the connection between general ML concepts and the limitations that propelled the field toward non-linear approaches such as Neural Networks (NN). Section 1.5 will instead be dedicated to showing how core concepts in ML training, such as underfitting, overfitting and validation procedures emerge naturally even in the simple Linear model setting.

The direct answer to some of these problems is introduced in Section 1.6. There, mainly traditional methods such as ridge regression and LASSO are quickly described. Naturally, the topic of regularization is huge and cannot fit a single section. Indeed, to some extent, this whole Thesis may be regarded as a travel between different Regularization and NN architecture choices that make them effective in a physics context. The topic will therefore be re-discussed at multiple points during the text.

One of the cornerstones of modern Machine Learning, the feedforward, Multi-Layer Perceptron (MLP) is presented, together with some fundamental mathematical results, in Section 1.7. Approximation capabilities will also be discussed and a simplified version of the Universal Approximation Theorem will be provided. Section 1.8 outlines how the optimization problem set by the loss function is tackled in practice. Some details on the backpropagation algorithm and how it is used in practice will be outlined.

Lastly, Section 1.9 discusses the role of symmetries in Machine Learning. Similarly to regularization, this is a huge topic: entire NN architectures have been developed to satisfy symmetry constraints. Again, in this Chapter we will only give some general hints on this topic, as the introduction of specific methods is better discussed contextually to applications in the following.

The Chapter cannot be an exhaustive presentation of Machine Learning and Neural Networks. Rather, it will hopefully just provide a quick and self-contained foundation for the terminology and general strategies that will be used in the rest of the Thesis. In general, we refer the reader who would like a more in-depth discussion to Refs. [21, 22, 23, 24, 25], which are the main source of information for this Chapter.

1.1 WHAT IS MACHINE LEARNING

Machine Learning (ML) has become ubiquitous in the last few years. Nowadays, terms like "Machine Learning", "Artificial Intelligence" and "Deep Learning" are everywhere in newspapers and scientific articles. Despite providing impressive results, the main idea behind most Machine Learning is quite straightforward: suppose we have couples of variables (x, y) (they could be scalars, vectors, words, images, etc.), and we know that variable y is correlated to variable x. This means that there is a function f such that:

$$y = f(x) \tag{1.1}$$

In the context of Machine Learning, x is usually considered as an "input variable" or a "feature vector", while y is often referred to as a "target", "label" or "output variable". Different terms come from the traditional distinction between classification (the task of assigning the correct label y to the features x) and regression tasks (the task of assigning the most appropriate output variable y to an input variable x, being x and y continuous values). In this Thesis, however, these terms will be used interchangeably, as happens in most modern ML. Indeed, in practice, the same methods can often be adapted for both contexts with minor modifications.

In most of this Thesis we will deal with so-called *supervised learning* [21, 22, 23, 25]. Informally, this can be described as the set of approaches to find a good approximation of the (unknown) function f. Indeed, ML reverses the standard scheme of computation: traditionally, we have some input for the calculations (e.g. initial conditions in a Differential Equations, x), and know the rules that will give the output (the Differential Equation itself, f). Then we perform some algorithm, either by pen and paper or by programming a computer, and obtain the solution of the equation (in our terminology, y). In supervised learning, on the other hand, we possess the input and the output, but we miss a representation of the mapping. The procedure of obtaining a "good" map (in a formal sense which will be defined in the next section) is called *training*. This kind of approach has proven to be particularly convenient for tasks in which formalizing the relationship between the x and y is hard. The typical example is recognizing cats from dogs: it's (usually) easy for humans to tell these two animals apart, but what are the exact features that allow for such a distinction?

This quality, however, is also very appealing for Materials Science, specifically if Machine Learning approaches are to be used in conjunction with experimental data: theory usually deals with idealized situations and reality is far more complex. ML could therefore provide an extremely flexible tool

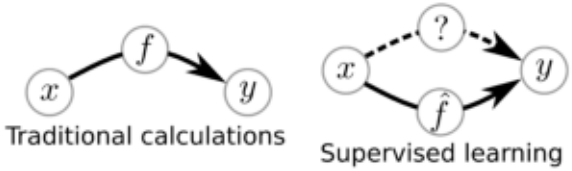

Figure 1: Comparison between traditional computation scheme and supervised learning. In the latter, we search for an approximation $\hat{f}$ for the expensive or unknown mapping between x and y.

in future applications where corrections terms to theoretical predictions are elusive. One of the main limitations, in this respect, is the requirement of building large datasets of experimental data.

The main advantage of ML (and in particular of Neural Networks) that will be exploited in this Thesis work, however, is the possibility of efficiently approximating complex nonlinear relationships between data. This is interesting from the computational Materials Science perspective because traditional methods are often computationally demanding or can be performed only under specific assumptions. Machine Learning can therefore represent a compact and general scheme to take the input of a traditional method x and find an efficient mapping f to obtain an output y which is a good approximation of the standard calculation. This scheme is recent but not completely original [24, 26], and has proven to be already very successful in the context of Molecular Dynamics [15, 27, 28, 29].

Up to this point, we have introduced just one type of Machine Learning. In reality, ML refers to a broader class of techniques. On the opposite side of the spectrum with respect to supervised learning, in which we want to train a machine to provide an output given an input, there is the so-called *unsupervised learning* [22, 23]. In this context, data are unlabeled, and statistical methods are used to find patterns. Examples of unsupervised approaches are inferring a probability distribution from data or finding correlations in unlabeled images. Naturally, the distinction between supervised and unsupervised is often blurry. Techniques such as conditional Generative Adversarial Networks (cGAN) [30], have input and output variables, but at the same time are trying to learn probability distributions. At the same time, methods such as Autoencoders [22, 23], a class of models trying to compress data, do not require labels but have an input and an output variable (although in this particular case $x = y$). The formation of such gray areas is somewhat promoted by the modular nature of many Machine Learning approaches, a point which will become more clear by the end of this Chapter.

For completeness, there is at least a third class of ML paradigms: Reinforcement Learning [31]. This is concerned, roughly speaking, with finding what is the most suitable interaction or action with a given environment. While there are very interesting applications for this class of algorithms in condensed matter physics [32], I have not used them during my PhD and they are therefore out of the scope of this Thesis.

1.2 SETTING THE GAME: PARAMETERS AND LOSS FUNCTIONS

Up to this point, the discussion has been very informal, with concepts like "appropriate output" and "close approximation" without a precise definition. We will now start to be more systematic. In this section, we will have in mind mainly supervised learning regression schemes. This is because they are usually more straightforward, closer to the typical material scientist work and more similar to most of the approaches presented in this Thesis.

First, let us reconsider the supervised learning task in a more mathematical framework. As stated in the previous section, most Machine Learning approaches involve the problem of approximating an unknown function f, given a set of its possible inputs $x = \{x_i, i = 1, ..., N\}$ and outputs $y = \{y_i, i = 1, ..., N\}$. Since the collection of (x, y) is used to train a model, it is usually called a *training set*.

We can formulate the training problem in a variational form: suppose that the function $\hat{f}$ is a candidate approximation of f. Then its outputs $\hat{f}(x_i)$ should be as close as possible to y_i. It is natural to consider the average sum of squared differences[1]:

$$\mathcal{L}[\hat{f}] = \frac{1}{N}\sum_{i=1}^{N}(\hat{f}(x_i) - y_i)^2 = \mathbb{E}(\hat{f}(x) - y)^2 \tag{1.2}$$

where $\mathbb{E}$ is the expectation value operator. Since $\mathcal{L}$ measures how good or bad an approximation $\hat{f}$ is, it takes the name of *Loss function*. The choice of squared differences instead of absolute differences is for the moment arbitrary. Indeed, different functional forms for $\mathcal{L}$ are often used in ML, depending on the task at hand. In principle, any convex function with a minimum in the origin could be used as a Loss function. In the next section, however, we will draw some connections with Bayesian methods in probability, which will highlight some rationale behind the choice of $\mathcal{L}$.

The concept of "best possible approximation" can then be formalized in the search for the $\hat{f}^*$ which minimizes $\mathcal{L}$. Of course, this search cannot be performed on the space of all functions. It is therefore convenient to consider a parametric class: $\hat{f} = \hat{f}(\cdot|\vartheta)$, where ϑ represent a set of (usually) real number parameters which prescribe how to map the input variable to the output one. Although it is not unique, this approach is very general and flexible. With the parametric form, $\mathcal{L}$ can be converted in a regular function of parameters:

$$\mathcal{L}(\vartheta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{f}(x_i|\vartheta) - y_i)^2 = \mathbb{E}(\hat{f}(x|\vartheta) - y)^2 \tag{1.3}$$

The training procedure amounts, in principle, to find parameters ϑ^* which minimize the loss function.

1 As $N \to \infty$, this is equivalent to considering the L_2 distance between functions.

1.3 PROBABILISTIC INTERPRETATION OF LOSS FUNCTION

Before moving to examples of Machine Learning concepts in practice, we will take a small detour on a probabilistic interpretation of the loss function and the optimization procedure of learning. In the following, we will consider familiarity with basic probability theory [20].

First, we need to introduce the concept of *parametric probabilistic model* [22]. A probabilistic model is, informally speaking, a set of assumptions which define a probability distribution relating variables. In our case, we will consider parametric models. This means that there exists a set of parameters ϑ which define the probability distribution of some variable. In concrete, we can regard the function approximation $\hat{f}(x|\vartheta)$ of the previous section as an object mapping an input variable x to the probability distribution of the output variable y.

Let us make this more concrete by considering a regression task in physical sciences: in general, experimental data are affected by noise, which is usually assumed to follow some Normal/Gaussian distribution $\mathcal{N}(\mu, \sigma)$, where μ is the noise mean and σ the standard deviation. The central limit theorem supports the ubiquitous presence of Gaussians. Suppose that x and y variables are measured and we are interested in finding the best approximation of the function f such that $y = f(x)$. To ease the notation, we will also assume that y is scalar, but generalization to vector-valued functions is straightforward. Under these assumptions, the actual realizations of y are independent and normally distributed around $f(x)$. For each couple i in the dataset:

$$y_i = f(x_i) + \varepsilon_i; \quad \varepsilon_i \sim \mathcal{N}(0, \sigma) \tag{1.4}$$

where we assumed that $\mu = 0$ and σ does not depend on the index i (homoscedasticity[2]).

With these assumptions, i.e. with this probabilistic model, the conditional probability of observing the values y given the input variable x and the set of parameters ϑ can be simply defined:

$$\mathbb{P}(y|x, \vartheta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{(y_i - \hat{f}(x_i|\vartheta))^2}{2\sigma^2} \tag{1.5}$$

On the notation side, in the following we will use capital letters for discrete probabilities, lowercase letters for probability densities and the calligraphic $\mathbb{P}$ when the distinction is not important. The term $\mathbb{P}(y|x, \vartheta)$ is called *likelihood*. Reverting to our original goal of "finding the best approximation of f", a reasonable alternative to the variational formulation might be to select parameters ϑ which maximize equation 1.5, a procedure called *Maximum likelihood Estimation* (MLE). In practice, it is equivalent but more convenient

2 Many methods can still be applied if this assumption is relaxed. Still, notation is more opaque, hence we will always assume homoscedasticity.

to consider minimizing the negative logarithm of the likelihood (as log is a monotonous function), as will become clear. Equation 1.5 becomes:

$$-\log \mathbb{P}(y|x,\vartheta) = \sum_{i=1}^{N} -\log \sqrt{2\pi}\sigma + (y_i - \hat{f}(x_i|\vartheta))^2 - 2\sigma^2 \tag{1.6}$$

Since the first and last terms do not depend on ϑ, this corresponds to minimize

$$\sum_{i=1}^{N} (y_i - \hat{f}(x_i|\vartheta))^2 = N\mathcal{L}(\vartheta) \tag{1.7}$$

which is equivalent to Equation 1.3 up to multiplicative constants. Notice, however, that the quadratic form of the loss function emerged automatically by the assumptions of the generative model: using a Mean Squared Error (MSE) loss function is equivalent to postulating that the data have a Gaussian distribution. Another interesting fact is that the actual value of σ does not influence the MLE. Indeed, as in most applications we will be interested in approximating *deterministic calculations*, we can imagine that $\sigma \to 0$, and the probabilistic formulation is a trick to get to rational Loss functions.

Up until now, it seems that we complicated the simple heuristic scheme which we introduced in the previous section. There are however at least three advantages that we have obtained from this more elegant probabilistic interpretation. First, we provided a rationale, based on statistical assumptions, on the specific forms the loss function should have. If, for example, we are developing a model of a Bernoulli random variable (which will be necessary in section 5), we now have the instruments to derive the correct loss function. Second, we have made an important connection with statistics: this means that we can exploit statistical methods to improve or to correctly interpret trained models. Third, this framework will be more flexible in the case the ML task is no longer to approximate a function, but instead it is to learn a probability distribution itself.

1.3.1 Bayesian interpretation

There is a further elaboration which we can do to obtain an even more flexible scheme for training. First, we need to review Bayes theorem:

Theorem 1. *Consider random events* X *and* Y*. Be* $\mathbb{P}(X)$ *indicate the probability of event* X *and* $\mathbb{P}(X|Y)$ *the conditional probability of* X *given* Y*. If* $\mathbb{P}(Y) \neq 0$*:*

$$\mathbb{P}(X|Y) = \frac{\mathbb{P}(Y|X)\mathbb{P}(X)}{\mathbb{P}(Y)} \tag{1.8}$$

The theorem is the same if all probabilities are conditioned on some third event Z*:*

$$\mathbb{P}(X|Y,Z) = \frac{\mathbb{P}(Y|X,Z)\mathbb{P}(X|Z)}{\mathbb{P}(Y|Z)} \tag{1.9}$$

Let us now shift perspective: while in the previous Section there was no uncertainty in the value of parameters (except for the fact that we were

searching for the optimal value), now parameters are themselves a random variable. Applying Bayes theorem, we can assign a probability to different sets of parameters:

$$\mathbb{P}(\vartheta|x,y) = \frac{\mathbb{P}(y|x,\vartheta)\mathbb{P}(\vartheta|x)}{\mathbb{P}(y|x)} \tag{1.10}$$

In Bayesian statistics, the term $\mathbb{P}(\vartheta|x,y)$ is called *posterior probability* and represents the probability we should assign to the parameters ϑ *after* having observed the dataset (x,y). On the right side of the equation, notice that $\mathbb{P}(x,y|\vartheta)$ is exactly the *likelihood*. The new term $\mathbb{P}(\vartheta|x)$ is instead called the *prior* and represents the set of previous assumptions we had on the values of the parameters. Instead of simple maximization of the likelihood, let us consider maximization of the posterior probability. This procedure is called *Maximum A Posteriori* (MAP) estimation. Similar to MLE, it is common to minimize the negative log-posterior. If we consider a normally distributed likelihood, we then obtain a new loss function:

$$\mathcal{L}(\vartheta) = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i - \hat{f}(x_i|\vartheta))^2}{2\sigma^2} - \log\mathbb{P}(\vartheta|x) \tag{1.11}$$

where we already got rid of all terms which do not depend on ϑ and normalized the summation. The first term in equation 1.11 is the usual MSE loss term used in regression, except for the σ term rescaling. There is however an additional prior term. For the moment, let us consider prior assumptions on the values of parameters that are independent of the training set observations, i.e. $\mathbb{P}(\vartheta|x) = \mathbb{P}(\vartheta)$.

The most general assumption is that we have no prior belief on ϑ: all values have equal probability[3], hence the prior term $\log\mathbb{P}(\vartheta) = \text{const}$. Since the prior term is not dependent on ϑ, then the posterior probability reduces to the likelihood term only: Maximum Likelihood Estimation can be recovered in the Bayesian picture as the special case of an uninformative prior.

Let us consider now another simple case in which ϑ are independent from x. If they follow a joint Gaussian probability distribution with zero mean, then

$$\mathcal{L}(\vartheta) = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i - \hat{f}(x_i|\vartheta))^2}{2\sigma^2} + \lambda||\vartheta||^2 \tag{1.12}$$

where we have already got rid of all terms not depending on ϑ again. The $||\cdot||^2$ term indicates the L_2 euclidean norm of ϑ treated as a vector of $\mathbb{R}^n$. For this reason, this term is called L_2 *regularization*. The meaning of this name will become more clear in section 1.5. λ is a so-called hyperparameter (to be distinguished from model parameters ϑ) balancing the effects of the likelihood and the prior term. Notice that this additional term in $\mathcal{L}(\vartheta)$ is pushing the optimal solution of the training procedure to have smaller parameters than the unconstrained procedure. In the specific example considered here,

3 Notice that, in principle, it is not possible to have a normalized uniform probability density on an unbounded region of $\mathbb{R}^n$. However, the assumption of such an *improper prior* is often performed in MAP because of its connection with MLE.

λ amounts to the ratio between the variances of the two Gaussian distributions: if this term is very small, we believe that the prior distribution is much broader than the model output one, hence the MAP will favor solutions which reproduce training examples almost exactly. If, on the other hand, λ is big, the optimal solution will have ϑ as small as possible, while the exact reproduction of the training examples will be less important. Notice that this moves the optimization procedure from the optimal value of likelihood. The reason why such a feature should be appealing will be discussed in Section 1.5.

Of course, normal prior is not the only convenient distribution. Another common choice is the Laplace distribution:

$$p_{\text{Lapl}}(x) = \frac{1}{2\alpha} \exp\left(\frac{|x-\mu|}{\alpha}\right) \tag{1.13}$$

If the position parameter $\mu = 0$, then the loss function is modified in the following way:

$$\mathcal{L}(\vartheta) = \frac{1}{N}\sum_{i=1}^{N} \frac{(y_i - \hat{f}(x_i|\vartheta))^2}{2\sigma^2} + \lambda|\vartheta| \tag{1.14}$$

where $|\cdot|$ indicates now the L_1 norm of ϑ. Again, $\lambda = 1/\alpha$ controls the relative importance of the two objectives. In general, one can consider priors belonging to the exponential family and obtain L_p regularization terms. Of course, there is no reason beyond practicality why every parameter should not have a different distribution and a corresponding individual λ term and we merely made this assumption here for ease of notation.

In some applications, the mathematical convenience of these priors also offers a nice interpretation (e.g. in linear regression they lead to ridge regression and LASSO). However, more general schemes can be considered. An example will be reported in Chapter 4, together with the relaxation of the usual assumption of independence between ϑ and x.

1.3.2 Prior alternative interpretations

There are at least two other interpretations of the prior term in the loss function. The first one is in terms of *penalty*: the prior term increases the cost (i.e. the loss) of some parameter choices. Indeed, it penalizes solutions with undesired properties, moving the optimal ϑ to a set of parameters which are more compliant with, for example, small L_p norms.

The second interpretation is in terms of *constrained optimization*: the hyperparameter λ can be regarded as the Lagrange multiplier associated with a constraint. In the case of L_2 regularization, we can rephrase the optimization task of equation 1.12 as finding the minimum-loss set of parameters ϑ contained in a hypersphere centered at the origin, with λ being (inversely) proportional to the radius of the hypersphere itself. This is a critical insight, as sets a correspondence between fixed parameters, i.e. infinitely strong constraints, and infinitely strong priors, i.e. certainty of some property. This will be useful when discussing specialized NN structures.

We close this section with a last key remark on prior distribution and the corresponding regularization theme: unfortunately, it introduces a new element in the optimization, the hyperparameter λ. If no other information is available, finding the best value of hyperparameters might be a non-trivial task: when models are simple and/or quick to train, this could be identified through a hyperparameter scan. In the case of more sophisticated models (e.g. Neural Networks), however, a systematic search could be too time or computationally demanding. Therefore, one has often to rely on trial-and-error procedures, which may miss optimal combinations.

1.4 LINEAR REGRESSION AS MACHINE LEARNING

We already introduced a lot of ingredients of modern Machine Learning. This is therefore a good point to stop and to see these concepts in action with a model familiar to all scientists: Linear Regression.

In linear regression, the relationship between the input variable x and the output variable y is mediated by a linear combination:

$$y_i = \sum_{i=1}^{N} w_i X_{ij} \tag{1.15}$$

the parameters w_i are called weights. The symbol w instead of the ϑ will be used specifically for linear models. Here we also introduced the *design matrix* elements X_{ij}, where row i corresponds to the $i-$th observation of the input variable $\vec{x}_i$. The index j, instead, runs on individual features of input variables. Simply put, the rows of $\overline{X}$ represent values for all the input variables in a single observation, while columns represent different values for the same feature across observations. As such, X_{ij} is the (scalar) value of feature j in observation i in the training set. Instead of linear transformations, we can consider the (slightly) more general case of affine mappings by augmenting $\overline{X}$ with an additional column[4] constant across observations and set to 1. If x is one dimensional, the coefficient related to this additional term represents the intercept value of the line approximating the (x, y) mapping and is usually called bias.

Linear regression can be generalized to encompass non-linear relationships between input and output variables. Indeed, the design matrix can be augmented with non-linear functions of the original input variables, under the constraint that the combination of such additional features is still linear. Linear models therefore include polynomial regression and expansion on basis functions. As an example, consider a third-degree polynomial fit:

$$\hat{y} = w_3 x^3 + w_2 x^2 + w_1 x + w_0 \tag{1.16}$$

4 This is the well-known trick of considering affine transformations in $\mathbb{R}^n$ as the projection of linear transformations in $\mathbb{R}^{n+1}$

Despite the relationship being explicitly non-linear, Equation 1.16 still is considered a linear model, as it is defined in terms of linear combinations of features, here represented by powers of x.

For simplicity, we are here assuming that the output variable is a scalar quantity. If we use matrix notation, a linear model tries to approximate the output variable vector $\vec{y}$ with the prediction $\hat{\vec{y}}$:

$$\hat{\vec{y}} = \overline{X}\vec{w} \tag{1.17}$$

Using the same intuition of section 1.2 (or using the MLE formalism), we can therefore search the "best" linear model as the minimum of the usual regression MSE loss function:

$$\mathcal{L}(\vec{w}) = \frac{1}{N}||\hat{\vec{y}} - \vec{y}||^2 = \frac{1}{N}||\overline{X}\vec{w} - \vec{y}||^2 \tag{1.18}$$

Minimization can be performed analytically by setting $\nabla_{\vec{w}}\mathcal{L}(\vec{w}) = \vec{0}$ and yields the MLE of the coefficient vector $\vec{w}^*$ in terms of the design matrix and the training output variables $\vec{y}$ only:

$$\vec{w}^* = (\overline{X}^T\overline{X})^{-1}\overline{X}^T\vec{y} \tag{1.19}$$

Equation 1.19 is commonly called *normal equation* or *least square solution*. In linear models, training amounts to a single, "one-shot" inversion of the $\overline{X}^T\overline{X}$ matrix, if possible. Indeed, matrix inversion could be ill-defined if the number of observations is too small with respect to the number of features in the input variable.

This can be understood from a statistical perspective. Let's go back to the design matrix itself, and consider its "centered" version $\overline{\mathcal{X}}$ such that:

$$\mathcal{X}_{ij} = X_{ij} - \mathbb{E}(x_j) \tag{1.20}$$

As this is just a constant shift for features, the least squares equation will lead to the same solution but for a modified bias term. Performing the same analysis of ordinary least squares, we have to invert the matrix $\overline{\mathcal{X}}^T\overline{\mathcal{X}}$, whose elements are:

$$(\overline{\mathcal{X}}^T\overline{\mathcal{X}})_{ij} = \sum_k \mathcal{X}_{ki}\mathcal{X}_{kj} = \sum_k [X_{ki} - \mathbb{E}(x_i)(X_{kj} - \mathbb{E}(x_j)] \tag{1.21}$$

If we remember that the first index in the design matrix runs on observations, we realize that this is just a re-scaled version of the covariance matrix of features in $\vec{x}$. This highlights that to have reasonable results with linear models, we should have a training set large enough to also have a reasonable estimation of covariance between features.

Another consideration comes from algebraic arguments. $\overline{X}^T\overline{X}$ is a symmetric square matrix whose rank is bounded by the number of observations: if the number of samples in the dataset is smaller than the number of features, then it is impossible to perform exact matrix inversion. While this fact may seem like a pathological case that is never encountered in the big data era, there are instead situations in which this may become relevant. Consider

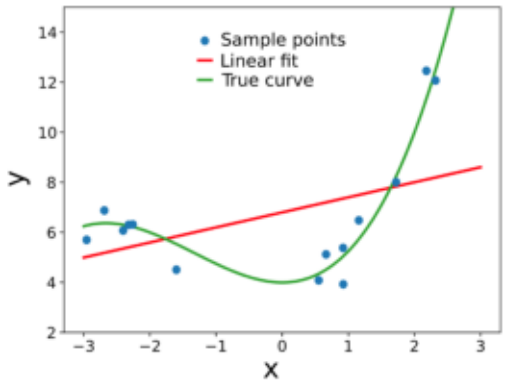

Figure 2: The "true" function $y = \frac{1}{4}x^3 - x^2 + 4$, shown in green, has been fitted using a polynomial of first degree $\hat{y} = w_0 + w_1 x$ (red line). The model is not flexible enough to capture non-linear relationships between x and y.

for example a general problem in computer vision, i.e. the identification of patterns in image data. The dimension of input variables N is therefore in the order of thousands of values even for moderate-resolution images. If now we use linear regression to approximate the $x \to y$ mapping with a polynomial function, we incur in the so-called *curse of dimensionality*: suppose we truncate the polynomial to degree k. If k is one, then we need to optimize $N + 1$ parameters (one coefficient for every feature plus a single bias term). For $k = 2$, input features are composed of pixel values, their square and all possible cross-products: a total of $N^2 + N + 1$ is required. As k increases, the number of parameters is $O(N^k)$ (at least in the most general case), which for large N may quickly exceed the number of examples in the training set. Analogous combinatorial problems can arise in other linear models, e.g. when considering Fourier basis expansion.

1.5 UNDERFITTING, OVERFITTING AND VALIDATION

There are four other very important concepts in ML: *underfitting*, *overfitting*, *regularization* and *validation*, which will now be discussed in the controlled environment of linear regression.

Underfitting refers to the phenomenon that can be observed when the model $\hat{f}$ which is used to approximate the real function f is too simple to reproduce trends present in the data. To show this in practice, we will quickly revisit the prototypical example of polynomial fit [21, 22]. Underfitting can be clearly understood if we try to fit a non-linear function using a line. Figure 2 shows a simple example of this: 15 points were extracted uniformly at random in the range $(-3, 3)$ and the function $\frac{1}{4}x^3 - x^2 + 4$ has been evaluated on those to construct a toy training set. Additionally, a noise term ε_i (with normal distribution with zero mean and standard deviation $1/2$) has been added to simulate experimental noise. Unsurprisingly, the result obtained shows that a line cannot approximate non-linear functions.

We already discussed, however, that a linear model can also reproduce non-linear relationships by augmenting the design matrix with, for example, powers of the input features. Indeed, as the degree of the polynomial in-

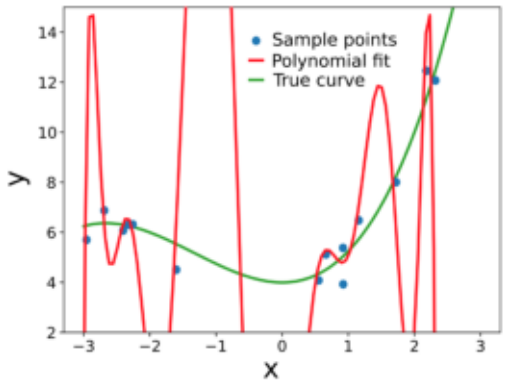

Figure 3: The "true" function $\frac{1}{4}x^3 - x^2 + 4$ has been fitted using a polynomial of degree 12, represented in red. Despite the fitted curve passing very close to all points, notice that the interpolation behavior oscillates wildly and is very far from the true curve (reported in green). This is an example of overfitting.

creases, the loss function calculated on the training examples will decrease (or at worst stay constant). In the limit, given N points (with distinct x values) there will be one and only one polynomial of degree $N-1$ which will pass through all of them, shrinking $\mathcal{L}$ to exactly zero[5].

If we consider the same example of Figure 2, but instead perform a fit with a polynomial of degree 12, we can observe that the curve now is closer to the training points (Figure 3). If we look at the comparison of the true (noiseless) function with the predicted one, strong oscillations between training examples can be observed. This is a phenomenon known as *overfitting*: the ML model is capable of closely reproducing the training set, but is terrible at *generalizing* on points which it was not exposed to.

While we have shown in practice overfitting, there is a general result for linear models, which we will prove for simplicity for scalar x and y here:

Proposition 1. *Consider a linear model obtained by the linear combination of* N *terms:*

$$\hat{y} = \sum_{i=1}^{N} w_i \psi_i(x)$$

where ψ_i *are arbitrary linear or non-linear functions (e.g. polynomials, Fourier components, Gaussians, etc.; mixed sets are also possible). Define the parameter vector* $\vec{w}_N = [w_1, w_2, ..., w_N]$ *and* $\vec{w}^*_N$ *as the parameters minimizing a loss function:*

$$\vec{w}^*_N = \text{argmin}_{\vec{w}_N} \mathcal{L}(\vec{w})$$

Then if we add a term $w_{N+1}\psi_{N+1}(x)$ *to the linear model:*

$$\mathcal{L}(\vec{w}^*_{N+1}) \leqslant \mathcal{L}(\vec{w}^*_N)$$

Proof. Given $\vec{w}^*_{N+1}$, by definition

$$\mathcal{L}(\vec{w}^*_{N+1}) \leqslant \mathcal{L}(\vec{w}_{N+1}) \quad \forall \vec{w}_{N+1}$$

5 This is true only if the design matrix stays well conditioned.

If instead we choose as parameter vector $\tilde{\vec{w}}_{N+1} = [\vec{w}^*_N, 0]$, the loss function will be higher or equal to the minimum one. Then

$$\mathcal{L}(\vec{w}^*_{N+1}) \leqslant \mathcal{L}(\tilde{\vec{w}}_{N+1}) = \mathcal{L}(\vec{w}^*_N)$$

Notice that the proof generalizes to any N by induction and that no assumptions on the functional form of the loss function have been made. □

This tells us that one should be careful to add parameters to a model: while adding terms to the linear combination will *always* decrease the loss function or keep it the same, this does not guarantee that the trained model will be useful in practice. And this is not even considering that the number of training examples conditions the rank of $\overline{X}^T\overline{X}$ or the curse of dimensionality!

Something is missing in what we have discussed so far: up until this point, we considered the best possible model as the one that minimizes the loss function on some training sets. Looking at the results in Figure 3, however, we are forced to change our perspective, as we want a ML model not only to reproduce the training set but also to provide reasonable predictions for unseen data. To assess this capacity, it is customary to split the available dataset into two subsets without overlaps: a *training set* and a *validation set*. The training set is used to optimize the parameters in the ML model, as we have seen in previous sections. The validation set serves as a proxy for estimating the *generalization error*: in practice, the loss function is calculated also on the validation set to monitor the overfitting phenomenon and to perform model selection. As an example, we reconsider a last time the polynomial fit case. In Figure 4 we show how the training and validation loss change with the degree of the polynomial. As it can be seen, the validation loss has a minimum: in some sense, there is an optimal polynomial degree which will give the best generalization results. Notice that this is not necessarily the degree of the (unknown) true polynomial. If, in any training algorithm, a diminishing of training loss is observed together with an increase of validation loss, overfitting is probably happening.

It is common in ML to discuss in terms of *model complexity* or *capacity*: very complex models (e.g. with a lot of parameters) will be capable of approximating highly non-linear relationships but are more prone to overfitting, as in the case of high degree polynomials. On the other hand, models with low "complexity" will not be able to capture all functional forms, but they will not incur in the risk of fitting noise or learning ill-behaved functions, like the simple linear model of Figure 2.

A more precise analysis in terms of *bias-variance trade-off* can be conducted. In this Thesis, however, we will not re-derive these results, as they are exact only for (generalized) linear regression. We therefore refer the interested reader to Refs. [21, 22, 24]. The main message, however, should be clear from Figure 4: just looking at the training loss is not enough for ML applications, and there is an optimal model "complexity" for generalization.

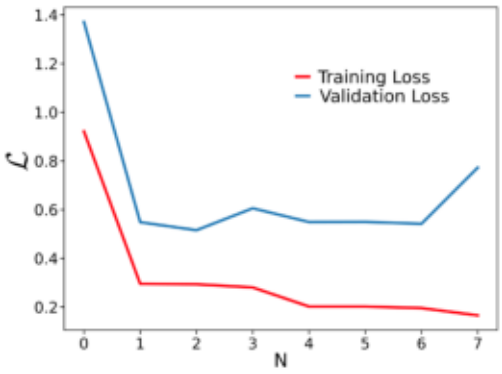

Figure 4: The "true" function $\frac{1}{4}x^3 - x^2 + 4$ has been fitted using a polynomials of degree N. The red curve represents the loss function as evaluated on the training set, while the blue curve represents the loss function on a validation set.

1.6 REGULARIZATION

It seems that the best way to tackle non-linear relationships is to add more and more terms to a linear combination of basis functions and monitor the generalization error through a validation set. However, the number of parameters cannot grow indefinitely, as overfitting and ill-conditioning of the normal equations bound their number. For this reason, it would be nice if there was a way to solve or at least alleviate these problems in a less trial-and-error way.

A powerful solution to this problem is *regularization*. On a high level, its effect can be regarded as a reduction in the *effective* number of parameters in a training procedure and solve some ill-conditioning problems. Let us for example the case of L_2 regularization in linear regression. The modified loss function is

$$\mathcal{L}(\vec{w}) = \frac{1}{N}[\|\overline{X}\vec{w} - \vec{y}\|^2 + \lambda\|\vec{w}\|^2] \tag{1.22}$$

We can proceed with basic algebra to obtain the corresponding modified normal equations for the minimum solution $\vec{w}^*$:

$$\vec{w}^* = (\overline{X}^T\overline{X} + \lambda\overline{\mathbb{1}})^{-1}\overline{X}^T\vec{w} \tag{1.23}$$

where $\overline{\mathbb{1}}$ is the identity matrix. Solution 1.23 is the same of equation 1.19 but for the addition of $\lambda\overline{\mathbb{1}}$ to $\overline{X}^T\overline{X}$. This modified linear regression is sometimes called "ridge regression". The main advantage is that this reduces ill-conditioned problems, as the presence of the identity makes matrix inversion easier and more stable from a numerical point of view. Some intuitions may explain why L_2 regularization should reduce overfitting. Model complexity is reduced, as the magnitude of ϑ is constrained: the class of functions with strong oscillations in between training examples is penalized. If we think back to the polynomial fit example, the optimal solution to the training procedure will have to find a compromise between high and low-order terms, thus reducing at least one of the overfitting sources.

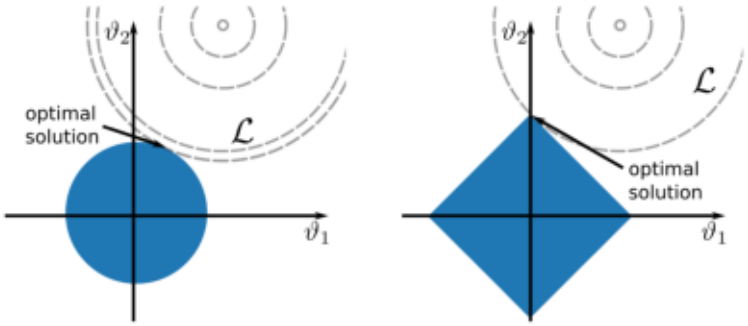

Figure 5: Classical picture showing the difference between L_2 (left) and L_1 (right) regularization. The area shaded in blue represents the set of valid parameters in the constrained optimization problem, while the dashed gray lines are contours of the un-regularized loss function in parameter space. In the L_1 case, diamond vertices can be solutions, implying sparsity.

As we have seen in section 1.3.1, another convenient regularization consists in the L_1 term $\lambda|\vartheta|$, which results in the so-called LASSO (Least Absolute Shrinkage and Selection Operator) regression. The analytic solution to the corresponding problem is not as straightforward as the L_2 or the unregularized one (after all, regularization does not have continuous derivatives). The different effect of the L_1 term with respect to the L_2 one can however be understood by a classical geometrical argument. If we consider the interpretation of regularization as constrained optimization, the L_2 term implies that parameters lie inside a N-dimensional sphere. On the other hand, L_1 regularization constrains parameters to be inside a diamond shape, as shown in Figure 5. For this 2D example, in correspondence with diamond vertices, the optimal solution implies that one parameter is set to zero. While ridge regression enforces small parameters, L_1 solutions enforce sparsity. This explains the presence of "Selection" in the acronym LASSO: depending on the value of λ, this method *selects* which terms in the linear combination are the most important. For example, if an odd function is expanded on sines and cosines, LASSO should detect automatically that the coefficients for cosines should be set to zero.

Of course, the effect will depend on the value of the regularization strength hyperparameter λ, which in the geometric picture is inversely proportional to the blue area. If its value is too small, the reduction in overfitting will be negligible, as we are solving the original unconstrained problem. On the other hand, if λ is too large, the model capacity will be so limited that the training procedure will not provide a good fit. In some sense, regularization is substituting the problem of finding which are the best basis functions with finding which is the optimal value of λ to avoid overfitting. This, however, can be solved utilizing the validation set: in the case of linear regression, a model is trained for different values of the hyperparameter. After that, predictions are compared with a validation set and the best value of λ is selected.

We should however be wary with this procedure as the validation set is in some sense now part of the training procedure. Since the chosen λ is minimizing the validation loss, we could incur in the risk of overfitting it to the validation set. For this reason, it is common in ML to have a third

dataset called *test set*, which is used as a proxy for real-world applications of the model.

While regularized linear models are a very powerful tool, we have not provided yet protection against the curse of dimensionality. Additionally, one would like to have a more general scheme for non-linear function selection than adding as many as possible and hoping that LASSO finds a sparse solution.

1.7 NEURAL NETWORKS

Up until now, we have only considered linear combinations as a means to approximate functions. In recent years, however, Machine Learning has relied more and more on non-linear approaches, the most famous of which are Neural Networks (NN). The idea behind NN dates back at least to the late fifties [33], and their name comes from connections with models for biological neuron systems. In this Thesis, however, we will consider NN "just" as a sophisticated Machine Learning method, which provides a flexible scheme for function approximation.

First, we need to define what a Neural Network is. To do that, we start with the simplest NN architecture, that of *single hidden layer, fully connected, feedforward Neural Network*. While this name may sound complex, from a mathematical perspective, it is very straightforward to define such a function:

$$\mathrm{NN}(\vec{x}) = \overline{W}_2 \cdot \sigma(\overline{W}_1 \cdot \vec{x}) \tag{1.24}$$

here, $\overline{W}_1$ and $\overline{W}_2$ represent matrices, $\vec{x}$ is the vector of input variables[6] and σ is a non-linear transformation, often called an *activation function*. When applied to vectors/matrices, non-linear functions are intended to operate component by component, i.e. for a vector $\vec{v} = [v_1, v_2, ..., v_N]^T$, $\sigma(\vec{v}) = [\sigma(v_1), \sigma(v_2), ..., \sigma(v_N)]^T$. $\overline{W}_1 \cdot \vec{x}$ is referred to as the hidden layer, as its values are normally only used in calculations and are therefore "hidden" in the NN black box (hence *single hidden layer*). Notice also that, depending on the number of columns of $\overline{W}_2$, Neural Networks can have vector-valued outputs. Figure 6 reports the typical graphical representation of this structure in the case of a vector input and scalar output. The *fully-connected* part of the name comes from the fact that $\overline{W}_1$ and $\overline{W}_2$ are dense matrices. If we consider their elements as "interaction weights", then all variables in $\vec{x}$ are interacting with each other. Finally, the term *feedforward* is used to describe operation flow: the simple NN we are discussing here has an output, which is calculated from the hidden layer, which in turn depends only on the input. At no point there is an inversion of this order. While this may seem like a useless specification at this point, the "feedforward" adjective is used to distinguish this ordered structure from more sophisticated architectures (see for example Recurrent NNs discussion in Chapter 4). In the language of NN, the dimensionality of $\overline{W}_1 \cdot \vec{x}$ is referred as the *number of neurons* or as

6 As in linear regression, $\vec{x}$ can be augmented with a "dummy" constant value so that matrix multiplication by $\overline{W}_i$ also encompasses affine transformations.

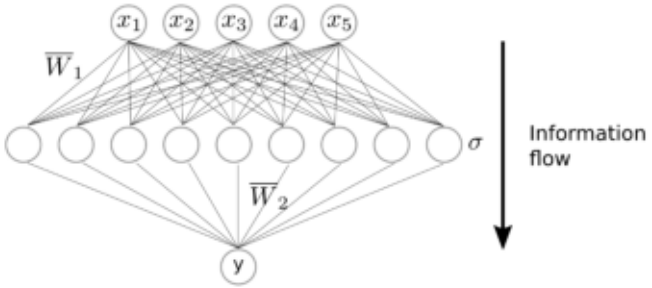

Figure 6: Graphical representation of a single hidden layer feedforward Neural Network. Lines represent multiplication by scalars. The NN is organized into three layers: an input layer (top), which receives input variables x (here 5D), a hidden layer (middle), obtained by the application of an affine transformation followed by a non-linear transformation, and an output layer (bottom), which can be regarded as a linear combination of the hidden layer activations.

the hidden layer *width*, while values $\sigma(\overline{W}_1 \cdot x)$ are sometimes called *neuron activation* or simply *activation*.

The goal of training a Neural Network is to find the optimal value (in the sense of training and validation scheme discussed for linear regression) of matrix elements in $\overline{W}_2$ and in $\overline{W}_1$, which constitute the set of parameters ϑ. Notice that, if we only consider optimization of values in $\overline{W}_2$, this is just a regular linear model. What makes NN special is that we are optimizing the linear combination term *and* what non-linear functions we are using at the same time: even though the functional form of σ is fixed, the presence of $\overline{W}_1$ modifies the specific function used based on training data. The capacity of a Neural Network can be increased by increasing the dimensionality of $\overline{W}_1 \cdot \vec{x}$, as we will proof in the following.

Traditionally, the non-linearity σ was chosen as a saturating function such as the hyperbolic tangent tanh or the so-called (logistic) sigmoid function:

$$\sigma(x) = \frac{1}{\exp(-x)+1} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{2}\right) \tag{1.25}$$

Notice that it is standard use to employ the symbol σ for both a generic non-linearity and the logistic sigmoid. When the difference is important, it will be clarified in the course of this Thesis. Unfortunately, σ is standard notation also for the stress tensor in linear elasticity (see Appendix B). To retain consistency with the literature, we have therefore chosen to use this symbol in both cases. The context should however allow the reader to identify the meaning: as a rule of thumb, in physical model equations σ is the stress field, while in NN architectures it represents a non-linear activation function. We will anyway try to be as clear as possible in the following.

Non-linear functions used in modern NNs encompass a very broad class of functions: in literature, applications of other bounded functions, unbounded and even non-smooth functions can be found. Maybe the most surprising activation function is the Rectified Linear Unit (ReLU):

$$\mathrm{ReLU}(x) = \max(0, x) \tag{1.26}$$

This freedom in the choice of σ may seem suspicious. Notice, however, that proposition 1 assures that if we increase the dimensionality of $\overline{W}_1 \cdot \vec{x}$, the loss function on the training set will decrease, as we are adding more and more terms to the linear combination mediated by $\overline{W}_2$. This is however a quite weak statement, as there is no guarantee that $\mathcal{L}$ will strictly decrease. In other words, we are still not sure that an arbitrary function can be approximated arbitrarily well by a Neural Network. Fortunately, *universal approximation* capabilities of NN have been proven under a very broad range of hypotheses. For instance, we report one of the classical theorem formulations by Cybenko [34].

Theorem 2. *Let σ be any continuous sigmoidal[7] function and $\vec{x} \in \mathbb{R}^n$ the input variable. Then, the finite sums of the form*

$$G(\vec{x}) = \sum_{j=1}^{N} \alpha_j \sigma(\vec{y}_j^T \vec{x} + b_j)$$

are dense on the set $C(I_n)$ of continuous functions on the unit interval $I_n = [0,1]^n$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(\vec{x})$, of the above form for which

$$|G(\vec{x} - f(\vec{x})|_\infty < \varepsilon$$

where $|\cdot|$ indicates the uniform norm $|f(s)|_\infty = sup(f(s)|s \in I_n)$.

The proof of this theorem requires results in functional analysis and is outside the scope of this Thesis. In the case of scalar functions, however, there is a simpler (and weaker) version of the universal approximation theorem[8]:

Theorem 3. *Given a primitive function f of a single variable $x \in [0,1]$, a series in the form G_N converges point-wise to f as $N \to \infty$.*

$$G_N(x) = \sum_{j=1}^{N} w_j \sigma(a_j x + b_j)$$

σ is a function which can approximate the Heaviside function[9] H

$$H(x) = 0 \text{ if } x < 0; \quad H(x) = 1 \text{ if } x > 0$$

Proof. We start by considering that the function f is primitive, hence it has a first derivative f' and can be written in the form:

$$f(x) = \int_0^x f'(y)dy = \int_0^1 H(x-y)f'(y)dy$$

7 Defined as a continuous function $\sigma(x) \to 1$ as $x \to \infty$ and $\sigma(x) \to 0$ as $x \to -\infty$

8 This was shown in the 2023 workshop in Roscoff GDR-IAMAT by Prof. Ludovic Goudenège; I take credit for possible unclarities or mistakes in the following

9 Being $x = 0$ a set of measure zero, the actual value of $H(0)$ is not important.

By hypothesis, H can be approximated by σ. For example, consider the logistic sigmoid:

$$H(x) = \lim_{\varepsilon \to 0^+} \frac{1}{1 + \exp \frac{x}{\varepsilon}} = \lim_{\varepsilon \to 0^+} \sigma\left(\frac{x}{\varepsilon}\right)$$

This means that by a suitable choice of parameter α, the following equation can be made arbitrarily close to the integral form of f:

$$f(x) \approx \int_0^1 f'(y)\sigma(\alpha x - \alpha y)dy$$

If integration is approximated by the series

$$f(x) \approx \int_0^1 \sigma(\alpha x - \alpha y)f'(y)dy = \lim_{N \to \infty} \sum_{j=1}^{N} \frac{1}{N}\sigma\left(\alpha x - \alpha\frac{j}{N}\right)f'\left(\frac{j}{N}\right)$$

Now set $\frac{f'(j/N)}{N} = w_j$, $\alpha = a_j$ and $-\frac{\alpha j}{N} = b_j$:

$$f(x) \approx \lim_{N \to \infty} \sum_{j=1}^{N} w_j \sigma(a_j x + b_j)$$

□

If instead of approximating H with σ, we integrate by parts the integral representation of f, the same steps yield the approximation theorem for ReLU networks, under suitable hypotheses. Both theorems 2 and 3 refer to unit intervals. Notice, however, that this generalizes to generic (closed) intervals by a simple rescaling of the input variables, which in principle could be learned by the Machine Learning algorithm itself.

Using the Neural Network slang, these theorems assure that a single layer NN can approximate arbitrarily well any continuous function as the hidden layer width increases. While the name *universal approximation theorem* sounds like a capital result, notice that similar results also hold for polynomial or Fourier series expansion. Why, then, NN are so successful with respect to more traditional approaches in linear models? The real difference between between linear regression models and NN is that in some sense they are optimizing the basis functions *and* the coefficients in a linear combination *at the same time*. This simple fact is very powerful, as it brings the solution to the curse of dimensionality. An intuition for this comes from simple mechanics as also illustrated in Figure 7.

Consider as an example the motion of a point mass attached to an anchor point, forming a pendulum. In general, the position of a point mass in 3D is described by three variables $[x, y, z]$. If however, we consider a *non-linear* combination of the input variables yielding the angle the pendulum describes with the vertical $\alpha = \arcsin \frac{\sqrt{x^2+y^2}}{z}$, then there exists a single degree of freedom describing the motion. The same kind of idea works also for other high dimensional data. A prototypical example (e.g. [22]) is that of photos: if we consider 100×100 images and turn on or off pixels with random intensity, it is very unlikely that we obtain an intelligible image. On

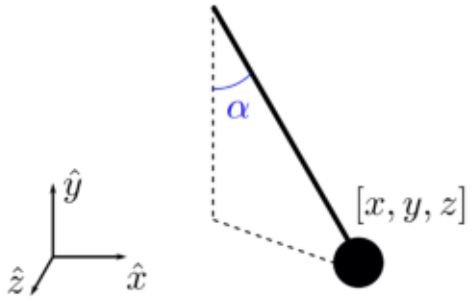

Figure 7: Example of dimensionality reduction from elementary mechanics. The motion of a pendulum in 3D should be defined in terms of the coordinate vector $[x, y, z]$. However, a suitable, non-linear transformation may be used to define a single scalar (the angle α), which completely determines the motion.

the other hand, a small perturbation of a real photo should yield something that is still recognizable. In other words, real data live on some (possibly complicated and non-linear) *manifold* of smaller dimensions in a very high dimensional space. In this picture, Neural Networks can therefore be regarded as a smart and automatic way to find a non-linear mapping from the input variables and such a manifold that is tailored to the task imposed by the loss function itself. To make another prototypical example, consider a classification task, in which high-dimensional feature vectors must be assigned to a class. From an intuitive point of view, NNs find the most efficient mapping from input variables to a non-linear manifold in which classes are easily separated (e.g. by a linear function).

To this point, we only discussed NN "width" as a means to obtain a universal function approximator. This class of architectures is also referred to as *shallow networks*. In ML, however, we often talk about *Deep* Learning. Indeed, there are dual results with respect to theorems 3 and 2 which assess that, if a network has a finite width, it is still capable of approximating functions (under regularity assumptions) arbitrarily well if multiple hidden layers are stacked, which is often referred to as increasing the Network *depth*. In this structure, the output of every layer is fed as the input to the next one. Such *feedforward, fully-connected Networks* can be described by the list of the subsequent layers width.

The possibility of having layered networks means that we can increase the representation capabilities of a NN either by increasing the dimensionality of the hidden layer (width) or by having more non-linear mappings increasing the network depth. In practice, the latter strategy is often more effective, even though there is no theoretical guarantee of which approach is the most efficient. Additionally, there is no requirement that different layers in a NN should be equal. This leads to a proliferation of different Network blocks that can be combined and stacked in multiple ways: one of the main reasons for Deep Learning's flexibility comes from its inherent modularity [22, 23]. Indeed, in modern Machine Learning computational blocks which were originally designed as NN on their own (such as encoder-decoder structures, recurrent blocks, etc.) became high-level blocks of very deep networks. Unfortunately, there is *no free lunch*: there is no clear winner in terms of NN

approaches. In practice, most ML choices are motivated by empirical testing and observation of what works best in the specific application at hand.

1.8 TRAINING A NEURAL NETWORK

Now that we have assessed that NNs have many interesting mathematical properties, we have to turn to the most problematic issue we have bypassed up to this point: Neural Networks are non-linear by construction, hence linear algebra cannot be used anymore to analyze their behavior and train them. Moving to the realm of non-linear functions comes with its price.

One of the most important effects is that the possibility of having general, closed-form solutions to optimization problems is lost: an equivalent to equation 1.19 is no longer available, hence iterative solvers have to be considered. One of the simplest ideas for minimum search of a function is the same as for *gradient descent*[10], a simple (local) minimization algorithm which is used in atomic configuration relaxation: given an initial condition (e.g. atom coordinates), forces are calculated through energy derivatives and atoms are moved proportionally in that direction. The proportionality factor is usually called *learning rate (lr)* in the ML community. The method proceeds in this fashion until a convergence criterion, usually on the force magnitudes, is met. If we substitute energy with $\mathcal{L}$ and atomic positions with parameters ϑ, this method could in principle be effectively used to train Deep Neural Networks.

In order to train a model through gradient descent an essential ingredient is an efficient way to obtain the full $\nabla_\vartheta \mathcal{L}$. Fortunately, NNs are composed of functions that are analytical and differentiable almost everywhere. It is therefore possible to calculate the gradient of $\mathcal{L}$ exactly, at least in principle. In modern Deep Learning, this is done through the so-called *backpropagation* algorithm. Despite its fancy name, this is essentially just an implementation of the Leibniz chain rule. To illustrate the main idea, let us consider a NN composed of N layers. To calculate $\mathrm{NN}(\vec{x})$, we have to traverse the so-called *computational graph* in the forward direction. In the simplest case of feedforward multilayer networks, for every layer we have to calculate the hidden activations $\vec{h}_i$:

$$\vec{h}_i = \sigma(\overline{W}_i \cdot \vec{h}_{i-1}) \tag{1.27}$$

where σ is the activation function as defined in the previous section, $\overline{W}_i$ represents the i-th affine transformation and $\vec{h}_0 = \vec{x}$. This passage is commonly referred to as the *forward* pass in the NN, as iterations start from the Network inputs and proceed towards its output, or, in the ML slang, from the input to the output nodes. Backpropagation efficiency comes from the

10 Also known as *steepest descent*, since movement in the configuration space is along the (negative) gradient direction, which corresponds to the fastest decrease in energy.

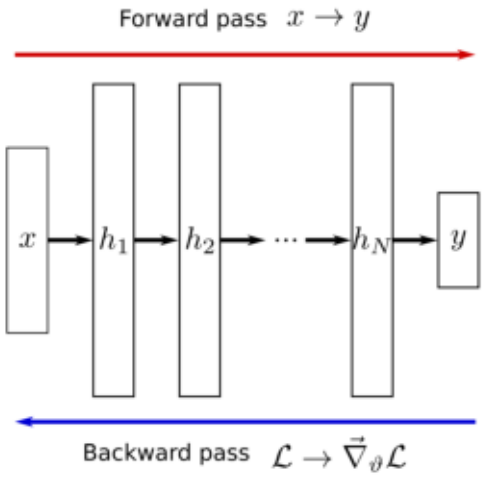

Figure 8: Sketch representing the forward and backward passages in NN training. The first refers to the mapping of x input to y output, while the second represents the inverse path used to calculate partial derivatives of the loss function with respect to the NN parameters.

realization that derivatives of the loss function with respect to parameters ϑ in $\overline{W}_i$ can be expanded by multiple applications of the chain rule as

$$\frac{\partial \mathcal{L}}{\partial \vartheta_j} = \frac{\partial \mathcal{L}}{\partial \vec{h}_N} \frac{\partial \vec{h}_N}{\partial \vec{h}_{N-1}} \frac{\partial \vec{h}_{N-1}}{\partial \vec{h}_{N-2}} \cdots \frac{\partial \vec{h}_i}{\partial \overline{W}_i} \frac{\partial \overline{W}_i}{\partial \vartheta_j} \tag{1.28}$$

Notice that the number of terms in the product in expression 1.28 increases the further we are from the NN output by going *backward* towards the input node (hence the name backpropagation). Remember also that NNs are composed of analytical and differentiable operations, which means that all the terms can be calculated explicitly. Additionally, intermediate partial derivatives are required multiple times (e.g. $\partial \vec{h}_N / \partial \vec{h}_{N-1}$ is used both when calculating $\partial \mathcal{L} / \partial \overline{W}_i$ and $\partial \mathcal{L} / \partial \overline{W}_{i-1}$), which means that computational efficiency can be greatly increased if track of these values is kept. This efficiency increase is not only a mere calculation speed-up: if numerical estimations, e.g. by finite difference, were used, the training procedure would become so slow and so affected by numerical errors (due to the composition of possibly hundreds of layers) that NN would only be theoretical mathematical objects. Modern computer libraries exploit dynamic programming to make backpropagation as efficient as possible even when custom layers are employed. I used one of those libraries, PyTorch [19], to write all ML codes used during my PhD[11].

Another drawback of NN with respect to linear models is that their inherent non-linearity removes all guarantees that the optimization task for the training loss is convex, i.e. the loss function will have multiple (local) minima in the space of parameters. Indeed, a trivial example would be swapping corresponding rows in $\overline{W}_1$ and $\overline{W}_2$: this will not change the output of the NN, but the two nets will potentially be very far apart in parameters space. As for all optimization tasks, the non-convexity problem should

11 Popular alternative frameworks are TensorFlow [35] and JAX [36]

sway us away from searching the global optimum, especially if the number of parameters is in the order of thousands (if not millions) and we lack any reasonable guess. In ML, however, we are often not interested so much in such a minimum, and we turn instead to a *good enough* set of parameters. While it is true that lower loss function models should perform better, there are at least two classes of arguments that suggest that searching for the exact global minimum is not so as critical as it seems: one is practical the other comes from overfitting.

1.8.1 Global minimum and convenient training

One possibility to approach the global minimum problem would be to ignore it. *In practice*, even though the optimization procedure converges to a local minimum, the solution found often provides reasonable generalization capabilities.

Of course, it would be nicer to have an explanation on *why* NN models not reaching global minimum perform so well. A good interpretation comes from integrating the discussion on the overfitting problem for NN. As we have seen in section 1.5, when the model capacity increases the possibility of obtaining a model that performs very well on the training set and extremely poorly in generalization also increases. Since NNs have universal approximation capabilities and it is not uncommon that the number of parameters used is in the order of millions, overfitting should be a main concern. Pretty much as in the case of high-degree polynomials, this phenomenon almost surely affects the global minimum of the loss function. In other words, remember that the real task of training is not to minimize the training loss as much as possible, but to obtain a model with as little generalization error as possible. In this respect, training is fundamentally *different* from pure optimization. In truth, most NN currently used not only are not at $\mathcal{L}$ global minimum, they are not even at a local one. This is due to the use of one of the most used regularization techniques, called *early stopping*. Basically, as iterative minimization of the training loss proceeds, validation loss is also calculated. As soon as the latter increases, gradient descent is stopped, as the model is starting to be affected by overfitting. It does not make sense to continue optimization, after all, as generalization capabilities (estimated through the *validation loss*) *degrade* as the procedure converges to a minimum of the *training loss*.

Before closing this Section, we will briefly discuss a suggestive analogy between Loss minimization and statistical mechanics. Let's reconsider MAP through Equation 1.10 (analogous considerations can be drawn for MLE by switching to an uninformative prior). The posterior probability $\mathbb{P}(\vartheta|x,y)$ can be rewritten as:

$$\mathbb{P}(\vartheta|x,y) = \frac{\mathbb{P}(y|x,\vartheta)\mathbb{P}(\vartheta|x)}{\int \mathbb{P}(y,\vartheta|x)\,d\vartheta} \tag{1.29}$$

The quantity at the denominator ensures that the probability distribution is normalized and is called the *partition function*. The fact that this name is the same as the normalization constant in statistical physics is not a co-

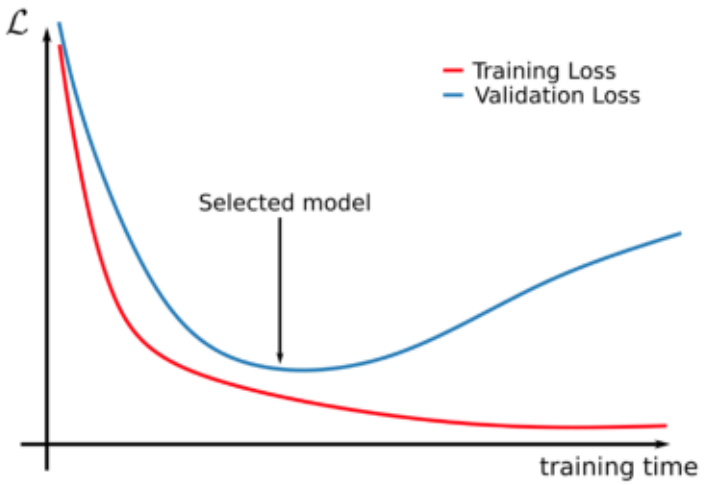

Figure 9: Illustration of the early stopping procedure. As the steepest descent proceeds, validation loss (blue curve) initially decreases, but at some point encounters a minimum, which corresponds to the NN with the best generalization capabilities. While the training loss (red curve) continues to decrease towards a local minimum, this does not correspond to a well-performing model.

incidence. In some sense, this is the same analogy we used in discussing gradient descent optimization: parameters ϑ play in NN the same role as atomic coordinates in molecular systems and $\mathcal{L}$ is analogous to energy. Finding the minima of the loss function yields the (locally) most probable model conditioned on the observations (x, y) pretty much in the same way as the minimization of energy yields stable configurations of a dynamical system. Indeed, (local) minima of the energy can be found without considering the partition function of the system and, while a lot can be extracted from these configurations, they do not tell all the physics. Searching for the global MAP estimation is equivalent to looking at the equilibrium states of a physical system. There is however a crucial difference: the MAP estimation can consider only a subset of all possible observations, i.e. those *in the training set*. This would be similar to minimizing the energy of a subsection of a physical system without considering interactions with the environment. It should therefore not be surprising that such a procedure can dramatically fail in generalization *even if a global minimum of the training loss is attained*.

1.8.2 Practical training

In this Section, we will briefly outline some features of training algorithms that are used in practice. We previously discussed how NNs need to resort to iterative optimization algorithms and sketched how the backpropagation algorithm can be used to efficiently implement gradient descent. This method, however, is rarely used by current ML practitioners. This is related to both theoretical and practical considerations.

The process of presenting the whole training set a single time to the NN is called an *epoch*. In practice, in one epoch, the Network produces the corresponding output variables for every input variable in the training set

once. Standard gradient descent has one evident disadvantage: since the loss function is some expectation value estimated on the whole training set, it performs a single optimization step in one epoch. This can make training extremely slow if the model considered is very big and the training set is very large. One way to speed up things is to consider that the expectation value can be calculated on a smaller subset of the data. This is the main idea behind *minibatch stochastic gradient descent*: at each epoch the training set is partitioned at random in sets of prescribed size, called *minibatches* or *batches*, and optimization steps are performed calculating $\mathcal{L}$ and gradients on the minibatch. The term *stochastic gradient descent (SGD)* is sometimes reserved for the limit case in which the batch size equals one, but in the following we will use the two names interchangeably. As the nomenclature suggests, this technique makes the optimization updates stochastic, as the gradient estimation will be affected by sampling noise. The smaller the batch size, the more noisy loss and gradient estimations will be[12]. While this may seem a big price to pay for a computational speed-up, there are many theoretical analyses proving that this procedure is indeed generating a Markov chain leading to a (local) minimum of the loss function [37, 38, 22, 39].

There are actually arguments for stochasticity being beneficial in training, in particular in avoiding shallow local minima and can be regarded as a regularization technique itself. A formal analysis of SGD characteristics is beyond the scope of the current Thesis, hence we refer the interested readers to Refs. [37, 22]. However, we will briefly discuss them from a heuristic point of view. While global minima of $\mathcal{L}$ do not necessarily yield well-generalizing models, it remains true that shallow, high-loss minima have a low likelihood and are not expected to provide good models. In standard gradient descent, where loss is calculated on the whole training set, avoiding such bad local minima is difficult: if the initial (random) set of ϑ is in the attraction basin of such a point, following gradients will result in a poorly performing model. If, on the other hand, SGD is used, there is a probability of escaping the basin by performing moves that are not aligned with the gradient or even in the opposite direction.

Of course, this is not the only available strategy to avoid bad local minima. In optimization literature, there are multiple techniques to modify the simple gradient descent method. From a ML perspective, two main approaches can be considered:

- *Gradient rescaling*: to speed up training in flat regions of the loss function landscape, the gradient is rescaled. At the same time, step size shrinks in high curvature regions. This is usually performed using some estimation of the gradient magnitude, the local curvature, or a combination.

- *Momentum*: similarly to damped motion in dynamical systems, information on the velocity at which the model is moving in parameters space is retained. This allows for easily surpassing shallow local minima (this effect combines with stochasticity in SGD) and speeds up

12 This is one reason why bigger minibatches are usually preferred to proper SGD with batch size one.

training in flat regions, but can slow down convergence in very steep regions. In any case, updates are no longer aligned with gradients.

Maybe the most successful algorithm in the last years is *Adam* [40], which combines both strategies. All training in this Thesis uses this method, which is reported as pseudocode in Appendix A. Some publications, however, suggest that Adam has shortcomings in some non-stationary problems [41, 42]. In the end, as for most things in ML, there is no clear best choice, however. For this reason, there is a plethora of optimization algorithms available in Deep Learning libraries and the related literature has substantially grown in the last decade [43, 44, 45, 46].

1.9 SYMMETRIES IN MACHINE LEARNING

One of the most important features in physical systems is the presence (or lack of) symmetries. For example, an invariant Lagrangian implies conserved quantities in the equations of motion [47]. It is therefore of utmost importance that whatever ML tool is used to approximate physical quantities complies as much as possible with known symmetries in the system. While this reason is certainly compelling, from the ML point of view there are other reasons why symmetries should have a role in ML model training.

From a loss minimization perspective, symmetries are constraints that the NN should satisfy. If prior knowledge of the symmetries of the function being approximated is inserted in the training procedure, this acts as a regularizer. From an intuitive point of view, this can be easily realized considering the following toy problem. Suppose that a polynomial regression is used to fit the harmonic potential energy of a particle. Setting the force constant to unity, the ground-truth expression would be $\frac{1}{2}x^2$. Of course, it is unreasonable that a training process would yield the exact form. Instead, the most probable estimate values for the polynomial coefficient will be returned. Based on its training examples, something like this could be produced:

$$f(x) = 0.50003x^2 + 0.0001x$$

Clearly, this expression will yield satisfactory results in terms of training loss, if x values in the training set are small enough. However, if this expression is used to integrate equations of motion, errors coming from the linear term will slowly start to accumulate. If, on the other hand, the symmetric property $f(x) = f(-x)$ had been enforced, the linear term would have been penalized[13].

Symmetry encoding in NN structures will be one of the leitmotivs of this Thesis. This is particularly critical in Deep Learning, as the huge number of parameters and flexible, non-linear nature of NN can make it non-trivial to insert symmetries. Several approaches have been designed for this purpose and many of them will be used in this Thesis.

13 Notice that this approach will not help to reduce the error on the coefficient for the quadratic term, however.

One of the most effective ways to encode system invariances in Neural Networks is to make their *input* itself invariant to system symmetries. Consider for example a potential energy function for a single particle in some external field. Naively, one would collect a training set of $([x, y, z], E)$ couples, relating the Cartesian coordinates of the particle to its potential energy. Suppose however that the physical system considered has a rotational symmetry. Then, the original training set could yield the same amount of information if instead of using $[x, y, z]$, the distance from the potential center $r = \sqrt{x^2 + y^2 + z^2}$ is used. This is very similar to what was discussed in section 1.7 about the existence of a non-linear, smaller dimensional manifold on which data live. Projecting input variables onto manifolds satisfying known symmetries can be regarded as making some of the NN "by hand".

In this respect, there is a second way to consider this kind of input variables transformation: the (non-)linear projection can be regarded as an additional, ad-hoc layer (with possibly ad-hoc activation functions) at the beginning of the Network. As transformations are fixed beforehand, parameters in this imaginary extra layer should be fixed as well. In other words, optimization algorithms do not affect their value. If we recall that loss function minimization comes from a MAP interpretation, this corresponds to an *infinitely strong prior*: we are *certain* that the parameters in this initial layer must be those set by hand because we are also certain that the input-output variable mapping satisfies certain symmetries. Of course, this certainty must come from some other sources than data. In statistical model terminology, assumptions used in modeling are called *inductive biases*. If inductive biases comply with the properties of the actual data source, then the resulting model will be more efficient (e.g. fewer parameters will be required, generalizations error will be lower, etc.).

An alternative approach to invariances is through a procedure called *data augmentation* [21, 22]. The main idea is to substitute the original training set with a symmetrized version of it. In practice, every time a data point is presented to the NN during training, a transformation of the input variables is applied. This is often performed in a randomized manner, with a similar spirit to minibatch gradient descent, which is particularly useful when the symmetry group acting on data is large. If the number of considered symmetries is small, on the other hand, input variables can be passed both in the transformed and in the "original" version. In some sense, data augmentation can be regarded as a way to artificially increase the training set size. Of course, these additional examples will have a different effect than collecting actual new data, as correlations are present with the original, non-transformed datapoints. The main advantage of data augmentation approaches is the fact that they are usually very simple and lightweight to implement. Finding symmetric representations, on the other hand, may be complex, as the underlying group structure may be fairly complex in real-world applications. The advantage is, however, that the NN output will be invariant to transformations *by construction*. While data augmentation will impose only *approximate* symmetries, compliance with input variable transformations is *exact*.

We close this section noticing that invariances are not the only symmetries present in data. Another important class is that of *equivariances*. These will

be discussed in Chapter 3 contextually to Convolutional Neural Networks, as they play a central role in image processing.

2 PREDICTING DISLOCATION INTERACTIONS BY ML

In this Chapter, we will apply the Neural Network theory described in Chapter 1 to a materials science problem: interacting dislocations in a strained SiGe film grown on a Si(001) substrate. This problem is technologically relevant, as dislocations are one of the main plastic relaxation mechanisms in semiconductor hetero-structures and their control is critical for the quality of electronic devices.

As we are interested in many dislocations interacting over several hundreds of nms range, atomistic approaches are out of discussion because of computational costs. Luckily, the physical modeling of dislocations in such systems can be done by standard tools in linear elasticity: dislocation-dislocation and dislocation-film interactions are described by the elastic energy stored in the associated stress and strain fields. As stated in the introduction, we will present physics contextually to the ML application. Linear elasticity represents an exception to this rule, as will be used on multiple occasions in this Thesis. For this reason, the main definitions and results are moved to Appendix B. There, the slightly less standard topic of eigenstrain formalism is also outlined and discussed. This general and flexible approach in linear elasticity allows for the numerical solution of elastic equations yielding stress and strain fields even for configurations where analytical expressions are missing, as in the presence of free surfaces near dislocations. The interested reader can find a more comprehensive treatment in Refs. [48, 49, 50, 51].

The Chapter is organized as follows. Section 2.1 provides a quick and qualitative description of flat SiGe heteroepitaxial films and the role played by dislocations in their relaxation. Basic dislocation theory is discussed in Section 2.2. As many ingredients are required to describe this class of defects in solids and their interaction with free surfaces, the section is rather lengthy. Readers familiar with dislocation theory can skip this section without loss of continuity. Extensions and more in-depth analyses can be found in [52, 53, 48, 54] and especially in [49].

Section 2.3 describes how ML approaches can be exploited to generate fast and accurate interaction potentials between general objects. This field of research has recently had a surge in popularity, especially in the context of Molecular Dynamics (MD) interatomic potentials. Here we will translate these ideas in the context of dislocation theory, showing that this can provide orders of magnitude speed-ups in elastic energy calculations. Section 2.4 describes how the training set is constructed. Moreover, it is discussed how force information can be used to obtain a more reliable approximation which allows for both statics and dynamics calculations.

Finally, Section 2.5 shows applications of the trained dislocation-dislocation and dislocation-strained film interaction potentials. In particular, minimum

energy configuration searches (static, thermodynamic calculations) and dislocation dynamics (kinetic processes) simulations will be shown.

Results presented in this Chapter can also be found in our publication [55].

2.1 DISLOCATIONS IN THIN FILMS

Dislocations are very important line defects in crystalline materials [49, 56]. They are a critical ingredient to describe the mechanical response of materials, both at the mesoscale and on a macroscopic level. In metallic systems, the formation, motion and interaction of dislocations is the main ingredient that allows for plastic behavior [49]. Their relevance, however, is not limited to structural properties. In semiconductor systems the presence and distribution of these defects must be closely controlled, as they may have severe impacts on the performance and properties of electronic devices [57, 56, 58, 59].

In this context, one of the most critical situations in which dislocation control is necessary is in heteroepitaxial growth. This process consists of growing a semiconductor film (called an epi-layer) on top of a crystalline substrate. In semiconductor technology, this is used for the integration of different materials on top of a substrate (e.g. in a heterojunction), or to exploit composition and structural effects to ameliorate electronic properties. The growth of Ge or SiGe alloys on pure silicon is one of the most studied systems because of its technological relevance [57]. In standard pressure and temperature conditions, silicon and germanium have the same crystal structure. Ge, however, has a larger unit cell by $\approx 4\%$ [60]. As incoming atoms have to accommodate in the substrate lattice position, the difference in lattice parameter induces a strain state in the epi-layer. Depending on the applications, relaxation of this strain may or may not be ideal, as deformations of the lattice cell have effects on the electronic properties of materials. As the film grows thicker, the elastic strain energy stored in the epi-layer increases, until it becomes energetically favorable to introduce dislocations in the system. As silicon and germanium yield an almost ideal solid solution, the same logic applies to the intermediate SiGe alloy. Indeed, the possibility of modulating the strain level in the epi layer with Ge concentration is one of the advantages of this system. A theoretical analysis of this process of strain energy accumulation, if the film morphology is flat, can be easily performed using the tools of linear elasticity.

Due to the substrate effect, volume elements of the epi-layer will be strained by:

$$\varepsilon = \frac{a_{Si} - a_{Ge}}{a_{Ge}} c_{Ge} = \varepsilon_{xx} = \varepsilon_{zz} \tag{2.1}$$

where a are lattice parameters and c_{Ge} is germanium concentration. Aligning the free surface normal to the y direction, ε will correspond to the strain state for both the xx and zz components. As there is no constraint along $\hat{z}$, the system is free to relax in the vertical direction. For these reasons, the described configuration is called a *biaxial strain system*. In the case of pure Ge, $\varepsilon \approx -4\%$.

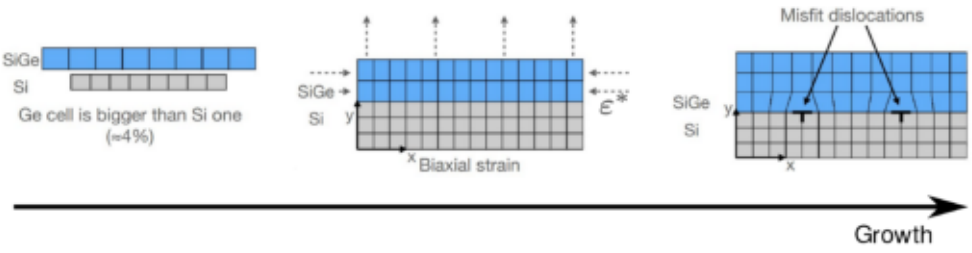

Figure 10: Schematic illustration for dislocation formation in growing SiGe layers. As the SiGe lattice cell is greater than the Si one (left), the film is initially coherent but strained (center). This leads to an accumulation of elastic energy, eventually making it energetically favorable to nucleate misfit dislocations (right).

Using the normal solid assumption, the strain energy density for a film containing a biaxial strain can be obtained as:

$$\rho_{el}(\bar{\varepsilon}) = \left[2(\lambda+\mu) - \frac{2\lambda^2}{2\mu+\lambda}\right]\varepsilon^2 = \frac{E}{1-\nu}\varepsilon^2 \tag{2.2}$$

where elastic constants E, ν, μ, λ are respectively Young modulus, Poisson ratio and Lamé constants as defined in Appendix B.

If the concentration of germanium c_{Ge} is constant, the elastic energy stored in a coherent (non-dislocated) film is linear with the film thickness W. Given the film surface area A, the total elastic energy E_{el} reads

$$E_{el} = A\int_0^W Y\varepsilon^2 dz = AY\varepsilon^2 W \tag{2.3}$$

As expected, the strain energy increases monotonically with the epi-layer thickness. This explains why dislocation nucleation in a growing film eventually occurs: as the thickness increases, the strain energy will at some point surpass the cost of inserting a defect with a strain field opposing the lattice mismatch. For thick films, multiple defects will nucleate, until the lattice mismatch is completely relaxed and the upper parts of the material will have restored their original lattice parameters. This relaxation mechanism is called *plastic relaxation*.

Notice that the derivation outlined here is valid only for flat morphologies. Indeed, the growth of SiGe films often does not proceed in this way, especially when germanium concentration and temperature are high and deposition fluxes are low (quasi-equilibrium conditions). Some details on growth mechanisms will be given in Chapter 3. For non-flat free surfaces a clean analytical solution is not available, however. The answer can be obtained numerically by Eshelby theory (see B.5), using the lattice mismatch as an eigenstrain term.

2.2 BASIC DISLOCATION THEORY

The linear elastic theory provides an effective tool to describe heteroepitaxial films. In this Section, we will outline the elementary theory of dislocations

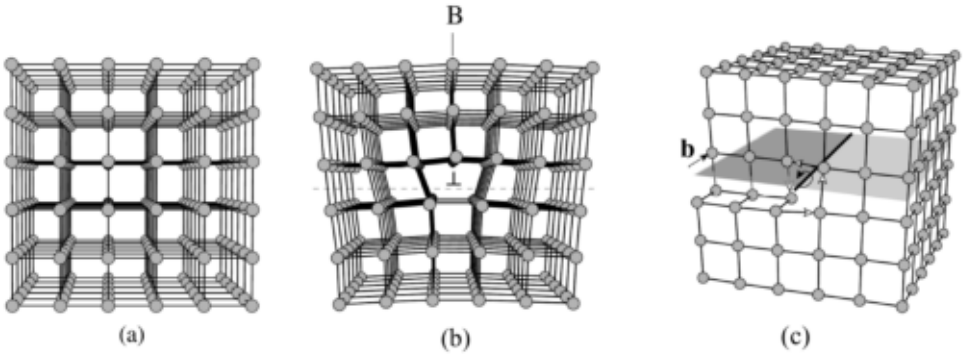

Figure 11: Atomistic view of dislocations. A perfect crystal (a) may be cut to insert an extra plane, giving an *edge dislocation* (b). cutting and slipping a surface with respect to the other gives a *screw dislocation* (c). Images adapted from [54].

in the same framework. This is convenient from at least three points of view. First, a continuum description is significantly cheaper from a computational point of view than an atomistic one. This is particularly important in the case of dislocations, as these defects are characterized by long-range interactions. Second and more importantly, it is well established that a description based on linear elasticity is effective in capturing dislocation behaviors, at least on the spatial scales considered in this Thesis [53, 49]. Third, elastic theory is elegant, compact and mathematically convenient. The total elastic energy can easily be decomposed in one-body and pair contributions using procedure B.35, simplifying the ML task. It is important to keep in mind that, despite these qualities, the calculation of stress and strain fields for an arbitrary dislocation and free surface configuration can be computationally challenging.

2.2.1 General concepts

Dislocations are *line defects* inside crystals. To describe their role, we shall follow a bottom-up approach, initially starting from an atomistic view and then moving toward a continuum description. Let us start by considering a perfect crystal, described by its lattice, as in figure 11. Now, imagine cutting the solid, removing or adding an atomic half-plane, and then gluing back the system. The resulting configuration is called an *edge* dislocation.

If instead of inserting an extra atomic plane we shear the cut half plane by a distance b, compatible with the crystal lattice, a *screw* dislocation is obtained. In general, a dislocation is not forced to have either a pure edge or a pure screw character. Mixed dislocations are possible, whose total stress field is given by a superposition of screw and edge components. The cut-and-gluing process defines a singular line, where the original, imaginary cut

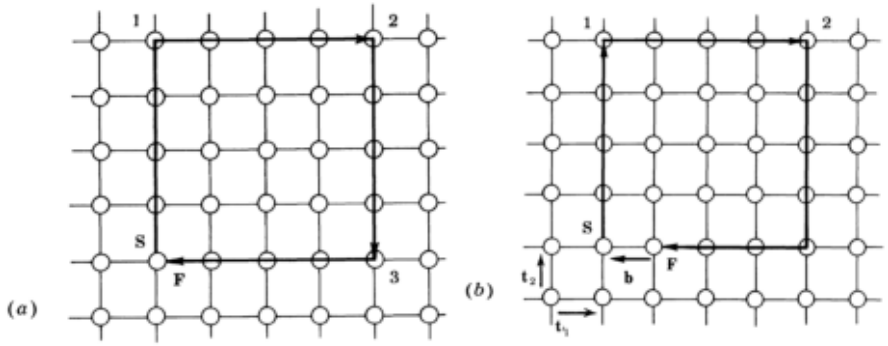

Figure 12: Burgers' circuit convention. The dislocation line is taken to be entering into the page. Adapted from [49]. For undislocated regions, $\vec{b} = 0$.

ends, that separates the "normal" part of the material from the "dislocated" one. This is called *dislocation line* and its direction will be indicated by the unit vector $\hat{\xi}$.

The character of a dislocation may be found through a geometric construction known as *Burgers circuit*. The idea is the following: starting from a lattice point in the crystal, we want to form a close loop[1] and count the number of "steps" that we take. As can be seen in figure 12, when the loop contains the dislocation line an additional step will be needed with respect to a loop in a "normal" part of the material. The vector closing the loop is called *Burgers vector* $\vec{b}$. Burgers vectors, therefore, represent the strength or charge of the dislocation, in analogy with electrostatics. From this discussion, it is also clear that the crystal structure and symmetries of the material will constrain possibilities for the values and directions of $\vec{b}$ (see 2.2.5). The continuum equivalent of the Burgers circuit is:

$$\vec{b} = \oint_{\Gamma} \frac{\partial \vec{u}}{\partial l} dl \tag{2.4}$$

where Γ is the line defining the circuit itself.

The screw or edge character of a dislocation can be established considering the angle between its line and its Burgers vector. Perpendicular configurations ($\hat{\xi} \cdot \vec{b} = 0$) yield an edge dislocation, while parallel configurations correspond to a screw dislocation ($\hat{\xi} \cdot \vec{b} = b$). Other cases yield mixed dislocations.

A main question remains when we are using a continuum theory for inherently atomistic objects. The main assumption of elastic theory is that deformations inside solids are small. However, we expect that the relative deformation (i.e. the strain) near the dislocation line will be in the order of unity. Indeed, linear elasticity is reliable *if and only if* we are far from the *dislocation core*, i.e. at points r such that $r \gg |\vec{b}|$). In fact, in linear elasticity stress and strain fields diverge near the core. These singularities are to be corrected either with a regularization procedure or by joining elastic calculations with *ab-initio* calculations on dislocation cores [61].

1 Both the choice of the dislocation line direction $\hat{\xi}$ and the direction in traveling around Burgers circuit are arbitrary. In this Thesis work, the right-hand rule will be used, so that if $\hat{\xi}$ is into the paper, the loop is clockwise.

2.2.2 Dislocations in bulk materials

Screw and edge dislocations can be characterized by their stress and strain fields. These can be obtained in a variety of ways, e.g. by defining the displacement field associated with the slip/atomic plane insertion or through the eigenstrain formalism. The interested reader can find step-by-step calculations in Refs. [53, 49].

Once dislocation strain or stress fields are known, it is possible to calculate the dislocation energy using equation B.37 or B.39. Given a general bulk dislocation with angle α between the Burgers vector and the dislocation line, by simple decomposition we have that

$$\frac{E_{mix}}{L} = \frac{\mu b^2[(1-\nu)\cos^2\alpha + \sin^2\alpha]}{4\pi(1-\nu)} \log\frac{R}{r_0} \tag{2.5}$$

in theory we should take limits $R \to \infty$ and $r_0 \to 0$. However the energy integral diverges in both cases. For the r_0 limit the reason is that we are trying to use an elastic theory's expression to calculate the energy near the dislocation line. A true elastic continuum would indeed require infinite energy to create such a core region, where stress and strain fields are singular. The cutoff radius r_0, therefore, indicates the distance at which elastic theory breaks down and we should resort to atomistic models. In principle, an extra term extracted by atomistic description should be added to expression 2.5.

An alternative approach is to employ a regularization procedure. A common practice is that of "smearing" the Burgers vector in the core region, to avoid singularities. In practice, instead of having a one-dimensional dislocation line, the Burgers vector is now distributed according to a "Burgers vector density" inside some tubular region. For calculations in this Thesis, we resorted to the regularization procedure described in reference [62], which allows for the removal of the ill-behaved regions if numerical methods are used to calculate dislocation stress/strain fields.

We will not analyze the $R \to \infty$ divergence, as it arises for infinite systems only and in the following we will be interested to dislocations near free surfaces [53, 49].

2.2.3 Free surfaces effects and forces on dislocations

The result derived in the previous Section is only valid for infinite straight dislocations in bulk. However real systems are never infinite. This is particularly important to consider in the applications we are interested in, as dislocations in epi-layers are significantly close to free surfaces. Stress and strain must therefore be calculated considering suitable boundary conditions, see Equation B.15. In particular, forces acting on free surfaces due to dislocations should vanish (traction-free condition). Analytical, closed-form solutions are available for *flat* surfaces and dislocations parallel to them. To simplify the mathematical treatment, the surface y coordinate will be 0, the film will be on $y < 0$, the vacuum on $y > 0$ and the dislocation core at y_0.

For screw dislocations the correction is straightforward. In analogy with electrostatic problems [63], a simple image construction suffices. Stress fields at the surface ($y = 0$) may easily be proven to vanish by direct calculation:

$$\sigma_{yz} = \left[\frac{b\mu}{2\pi}\frac{x}{x^2+(y-y_0)^2} + \frac{(-b)\mu}{2\pi}\frac{x}{x^2+(y+y_0)^2}\right]\Big|_{y=0} = 0 \tag{2.6}$$

the equivalent result also holds for the only other non-vanishing component σ_{xz}. Edge dislocations are more complicated. An analytical solution, which is composed of an image construction and an additional correction, has however been found by Head [64].

We directly report the form for a mixed dislocation:

$$\frac{E^{surf}_{mix}}{L} = \frac{\mu b^2[(1-\nu)\cos^2\alpha + \sin^2\alpha]}{4\pi(1-\nu)}\log\frac{2l}{r_0} \tag{2.7}$$

being $l = \sqrt{|y_0|^2 - r_0^2}$ and α the angle between $\vec{b}$ and $\hat{\xi}$, again.

If $|y_0| \ll r_0$, we can identify l with the dislocation distance from the free surface. In this limit, expression 2.7 can be used to calculate the *force per unit length* that acts as an effect of the surface on the dislocation:

$$\frac{F^{surf}_{mix}}{L} = \frac{\partial}{\partial l}\frac{E^{surf}_{mix}}{L} = \frac{\mu b^2[(1-\nu)\cos^2\phi + \sin^2\phi]}{4\pi(1-\nu)l} \tag{2.8}$$

this force is directed towards the surface, since a perfect solid *tends to expel dislocations*, as can be clearly seen with the energy of dislocation increasing with l. If an additional stress field is present in the material, however (e.g. due to lattice mismatch in a heteroepitaxial film), dislocations may have an equilibrium position inside the body.

Equation 2.8 can be verified to be a special case of the so-called Peach-Koehler force due to the dislocation *self stress*, i.e. the stress generated by the dislocation itself at its core. This term is not present in bulk systems but arises from the traction-free conditions for dislocations near free surfaces.

For generic stresses, the force per unit length acting on a dislocation reads:

$$\frac{F}{L} = (\overline{\sigma}_{tot}\cdot\vec{b})\times\hat{\xi} \tag{2.9}$$

This formula is equivalent to the derivative of the total dislocation energy with respect to dislocation configuration and in the simple, infinite, straight line configuration considered so far, reduces to a function of the dislocation core position only.

2.2.4 Periodic Boundary Conditions

Up to this point, we treated isolated dislocations in (semi)infinite materials. In computer simulations, however, it is often computationally advantageous to consider Periodic Boundary Conditions (PBCs). In terms of dislocation configurations, this corresponds to periodic distributions of defects in a film with a periodic profile. Indeed, this allows one to describe the behavior

of an effectively infinite number of dislocations. Of course, such boundary conditions should be treated with care, since dislocations in the neighboring cells are not independent. This means that any observation of behaviors whose wavelength is larger than the simulation cell is lost. Notice, however, that in heteroepitaxial films is indeed possible to observe periodic patterns of surface morphology and dislocations [65, 66], which justifies the choice of PBCs to some level.

To avoid the energy divergence problem at cores, a regularization procedure like in [62] will be used in the following. The corresponding stress functions can be summed analytically to give a close expression for an infinite array's fields. These results can be derived from Ref. [65] or can be obtained as in [67] and are reported in Appendix C. These are important, as they can be used as an eigenstrain term for dislocations near an arbitrary, undulated surface.

2.2.5 Dislocations in SiGe system

As already noticed, Si and Ge both crystallize in the same structure. In particular, they take the diamond structure, which can be described as an FCC lattice with basis. In such crystals the $\frac{1}{2}\langle 110\rangle$ (in lattice constant units) translation vectors for the lattice are stable Burgers vector [49]. Indeed, these are the shortest lattice vectors and the dislocation energy scales as b^2.

Since stress and strain fields for dislocations whose line is parallel to the z axis are already available, in simulations we will rotate the frame of reference accordingly by 45° around the y axis, which is aligned with the (001) crystalline direction and points out of the free surface. If we only consider Burgers vector relaxing the heteroepitaxial misfit, we have:

$$\vec{b}_1 = b\begin{pmatrix}\frac{1}{2}\\ \frac{1}{\sqrt{2}}\\ \pm\frac{1}{2}\end{pmatrix} \qquad \vec{b}_2 = b\begin{pmatrix}\frac{1}{2}\\ -\frac{1}{\sqrt{2}}\\ \pm\frac{1}{2}\end{pmatrix} \qquad \vec{b}_3 = b\begin{pmatrix}1\\ 0\\ 0\end{pmatrix} \tag{2.10}$$

The stable nature of these defects is made manifest by the PK force caused by the residual misfit stress being in the negative y direction, as can be confirmed by direct calculations. These dislocations are usually classified based on the angle between $\vec{b}$ and the dislocation line ξ. For this reason, $\vec{b}_1$ and $\vec{b}_2$ defects are referred as 60° *dislocations*, while $\vec{b}_3$ defects are called 90° *dislocations*

If we further restrict to the (x, y) plane, z component of $\vec{b}$ is no longer relevant and we are left with three choices:

$$\vec{b}_1 = b\begin{pmatrix}\frac{1}{2}\\ \frac{1}{\sqrt{2}}\end{pmatrix} \qquad \vec{b}_2 = b\begin{pmatrix}\frac{1}{2}\\ -\frac{1}{\sqrt{2}}\end{pmatrix} \qquad \vec{b}_3 = b\begin{pmatrix}1\\ 0\end{pmatrix} \tag{2.11}$$

2.2.6 Dislocation motion

Let us now briefly outline dislocation motion. When a dislocation travels through the crystal lattice, we can decompose its movement in two com-

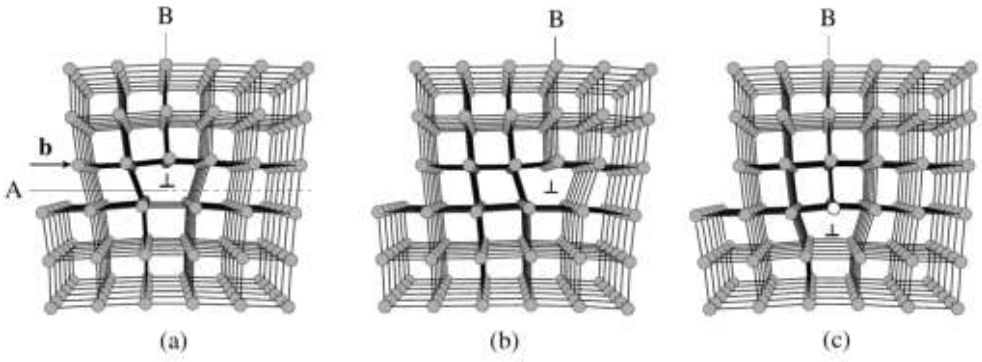

Figure 13: Difference in glide and climb motion for an edge dislocation. The extra plane is indicated by the letter B, while the glide plane is indicated by the letter A. Starting from the original position (a), it is possible to see that the glide motion (b) movers the extra plane and conserves the total number of atoms, while climb motion (c) requires an extra row (painted in white). Image adapted from [54].

ponents, the *glide* motion and the *climb* one. These are best seen for edge dislocations.

Glide motion happens along the plane containing both the dislocation line and the Burgers vector, i.e. the plane normal to $\vec{\xi} \times \vec{b}$. This plane is called the *glide plane* for the dislocation. As is depicted in figure 13, one can see that glide motion is the easier one, since it requires the breaking and re-forming of only one atomic bound at a time.

On the other hand, the climb motion is perpendicular to both the dislocation line and the glide plane. This motion has a high activation energy since it corresponds to the condensation (or emission) of vacancies around the dislocation line. For this reason, climb motion is also called *non-conservative*, while glide is *conservative*.

Following these arguments, one may expect that a dynamical description of dislocation motion should consider primarily glides. This is indeed true [49, 54], and in Dislocation Dynamics simulations climbs are often disregarded unless high temperature (hence high vacancies concentration and available thermal energy) is achieved. Considering SiGe systems and dislocation lines parallel to the substrate-film interface, the glide plane of 90° dislocations is also parallel to the SiGe/Si(001) interface. The stress field generated by lattice mismatch, however, generates a PK force that is directed downwards and normal to the glide plane, hence this class of dislocations is usually considered *sessile*. In other words, should a 90° dislocation form, it will be pinned. 60° dislocations, on the other hand, do not have this constraint.

From an experimental point of view, it is well established that dislocations in SiGe films are mainly 60° dislocations [58, 57]. Notice, however, that sessile defects may form as a result of *dislocation reaction* [53, 49]: if a $\vec{b}_1$ and a $\vec{b}_2$ dislocation encounter, they may combine. The resulting dislocation has a new Burgers vector given by the sum $\vec{b}_1 + \vec{b}_2$, hence a 90° $\vec{b}_3$ dislocation is obtained. As $\vec{b}_1$ and $\vec{b}_2$ dislocations can perform such combination, we will call them *complementary* in the following. Naturally, not all dislocation reactions are possible or energetically favorable. Within dislocation theory

an energetic principle called Frank criterion [53, 49] is often invoked, establishing under which circumstances dislocation merging/splitting may happen. In this respect, $\vec{b}_1$ and $\vec{b}_2$ reaction is energetically convenient. Notice, however, that this does consider other stress fields possibly present in the material. Indeed 90° dislocation formation will always be observed: in Section 2.5 we will show that for deviations from flat profiles, 60° dislocations reactions may be hindered.

2.3 DEEP LEARNING POTENTIALS

As already pointed out, analytical expressions for dislocation deformation fields are available only for very simple configurations in epi-layers (e.g. straight lines and flat free surfaces). While many interesting behaviors can be analyzed under these assumptions [68, 69, 70, 71, 72], their limitations are evident. In particular, the requirement of a flat free surface hinders the study of dislocation thermodynamics and kinetics in many technologically relevant configurations, such as nanostructures and corrugated films.

In recent years, the research group I worked with during my PhD tried to overcome these limitations in several contexts. These studies retain as a simplifying assumption parallel, straight dislocation lines but relax the constrain of flat surfaces [73, 74, 75]. This allows one to study complex surface morphologies while containing simulation costs, as the computational domain is 2D. The main computational tool is the numerical solution of the elastic mechanical equilibrium by Finite Element Method (FEM) calculations. This approach is particularly well suited for dislocation problems, as the computational domain can be discretized with adaptive meshes, yielding a higher accuracy to computational costs ratio. FEM meshes, indeed, can be easily refined near dislocation core regions, where stress/strain fields fluctuate rapidly even if regularization approaches such as the one discussed in 2.2.2 is used. Dislocation fields are obtained using analytical solutions as eigenstrains and allowing the solver to find displacements complying with the free-surface boundary conditions. A more complete description of this approach can be found in Refs. [76, 75]. The FEM output of a typical calculation is shown in Figure 14.

Despite the advantages of FEM mesh refinement, this approach is however convenient only for configurations containing few dislocations or for one-shot calculations of elastic energies. For more demanding scenarios such as Monte Carlo searches, energy minimization, or simply calculations involving a high number of dislocations, the computational costs of FEM are still a hard bottleneck. On the other hand, the number of degrees of freedom required to fully characterize a dislocation configuration in a fixed film geometry is relatively limited. Indeed, a list of the Burgers vector components $\{\vec{b}_i\}$ and the positions of dislocation cores $\{\vec{r}_i\}$ are sufficient to fully describe the system state. Hence, there exists, at least in principle, a function $\mathcal{E}$, such that the elastic energy of the system E is

$$E = \mathcal{E}(\{\vec{r}_i\}, \{\vec{b}_i\}) \tag{2.12}$$

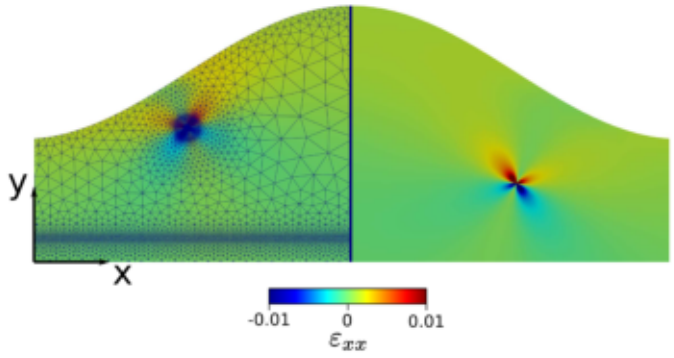

Figure 14: Example of a strain map obtained by FEM calculations. The xx component of the strain field is reported. Dislocation lines $\hat{\xi}$ are exiting from the paper plane. On the left side it is possible to observe the FEM mesh used in calculations. Notice the refinement near dislocation cores.

Indeed, the FEM solver is just a computationally expensive tool to evaluate $\mathcal{E}$.

The idea of surrogating a computationally intensive energy function with a Neural Networks[2] actually traces back at least to Blank et al. in 1995 [79]. The main idea was to fit energy terms coming from *ab-initio* electronic calculations for H_2-surface systems. Once the potential energy function is available, Molecular Dynamics (MD) simulations can be sped up using the faster NN approximation. Indeed, in the development of what are now called Deep Learning potentials [15, 80, 81], the problem is very similar to the dislocation one we are considering in this Chapter: the potential energy of the system is parametrized by a reduced number of variables (nuclear coordinates and species), but the proper way to calculate its value involves solving a complex many-body quantum mechanical problem.

2.3.1 Decompositions and symmetries in Deep Learning potentials

Initial works on Deep Learning potentials had two major limitations [80]. First, as fully-connected, feedforward NN was used, the application of the trained model was limited to a system containing the same number of particles. Suppose that a NN is trained in the most naive way, mapping the set of 3N cartesian coordinates of the N particles in the training set simulation cell to a single scalar, the potential energy. Then, the first layer of the NN as presented in Section 1.7, will be an affine transform from $\mathbb{R}^{3N}$ to $\mathbb{R}^{M}$, where M is the first hidden layer width. If a potential is required for configurations in which one atom is added or subtracted from the system, the whole training is to be done again from scratch.

This problem can be addressed by decomposing the total energy in terms of *local contributions* which can be ascribed to each atom. While in electronic *ab-initio* calculations there is no such thing as "the energy of atom i in the simulation cell", in Deep Learning potentials it is supposed that the total energy of the system can be expressed as a summation of such contribu-

2 Notice that there are also alternative ML methods to approximate MD interatomic potentials. See for example [77, 17, 78]

tions [80]. This assumption is not a new introduction into Deep Learning potentials: indeed, classical, semi-empirical potentials make a similar assumption [82]. In practice, for a system of N particles:

$$E_{tot} = \sum_{i=1}^{N} E_i(\vec{x}_i^{env}) \tag{2.13}$$

where E_i represent said local contributions and $\vec{x}_i^{env}$ is some *local environment description* of particle i surroundings. In principle, this can be as simple as the list of coordinates for nuclei in some suitable neighborhood of the particle.

The second main problem comes from the lack of symmetries in the learned potential energy approximation. As briefly discussed in Section 1.9, symmetries play an essential role in regularizing machine learning models. NN functions used to approximate interatomic potentials should encode, for example, translational invariance, so that the generated (microcanonical ensemble) dynamics obey momentum conservation. Another important symmetry for potential energy functions in molecular dynamics is the exchange one: if atoms i and j of the same chemical species are swapped, then the total potential energy remains unchanged. Of course, this should hold also for the local contributions of Equation 2.13. These sorts of symmetries are not *a-priori* present in fully connected, feedforward Networks. For DL potentials, physical symmetries are usually introduced via input transformations, yielding so-called *atomic environment descriptors* [15, 83].

2.4 TRAINING NNS FOR DISLOCATIONS

2.4.1 Energy decomposition

As we have just seen in Section 2.3.1, writing the energy of dislocations as a summation of individual contributions would allow for generalizations beyond the number of defects in the training set.

Using basic linear elasticity results, it can be shown that the total energy of a system of dislocations E_{tot} in a strained film can be *exactly* decomposed into self-energy and pair terms. In particular:

$$E_{tot} = \sum_{i}^{N} V_i + \sum_{i}^{N} H_i + \frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{N} W_{ij} \tag{2.14}$$

Indeed, from the linear theory of elasticity, the energy of a (traction-free) deformed body Ω can be expressed as (see Appendix B and Refs [53, 49]):

$$\begin{aligned} E_{tot} =& \frac{1}{2}\int_{\Omega} \sigma_{het} : \varepsilon_{het} d\vec{x} + \sum_{i}^{N}\int_{\Omega} \sigma_{het} : \varepsilon_i d\vec{x} + \frac{1}{2}\sum_{i}^{N}\int_{\Omega} \sigma_i : \varepsilon_i d\vec{x} + \\ & \frac{1}{2}\sum_{i}^{N}\sum_{j\neq i}^{N}\int_{\Omega} \sigma_i : \varepsilon_j d\vec{x} \end{aligned} \tag{2.15}$$

where "het" subscript indicates fields generated by the heteroepitaxial mismatch and i and j index dislocations.

Physical interpretation of terms in Eq. (2.15) is as follows. The first term represents the elastic energy stored in a mismatched, fully-coherent film. For a fixed surface morphology this is a constant and can therefore be neglected for our goals. The second and third terms represent the dislocation-heteroepitaxial film interaction and the dislocation self-energy, i.e. the distortion energy associated with an isolated dislocation. In the case of a flat surface, this term amounts (per unit length) to Equation 2.7, if no PBCs are used. Even in the case of undulated surfaces, the value of these terms only depends on the position and Burgers vector of dislocation i. They are in other words, *one body terms*. The last term is the dislocation-dislocation interaction energy, only depending on the positions and Burgers vectors of each pair. Notice also that, by symmetries of the elastic tensor $W_{ij} = W_{ji}$, i.e. dislocation-dislocation interactions are symmetric, as should be expected by the action-reaction principle.

Comparison between Eq. (2.14) and Eq. (2.15) yields:

$$\begin{aligned} H_i &= \int_\Omega \sigma_{het} : \varepsilon_i d\vec{x} = H_i(\vec{r_i}, \vec{b_i}) \\ V_i &= \frac{1}{2}\int_\Omega \sigma_i : \varepsilon_i d\vec{x} = V_i(\vec{r_i}, \vec{b_i}) \\ W_{ij} &= \int_\Omega \sigma_i : \varepsilon_j d\vec{x} = H_i(\vec{r_i}, \vec{r_j}, \vec{b_i}, \vec{b_j}) \end{aligned} \tag{2.16}$$

where explicit dependence from dislocation positions $\vec{r}$ and Burgers vectors $\vec{b}$ has been added. To approximate the total energy of a system containing *an arbitrary number of dislocation*, it is therefore sufficient to build three NN models, one for each term in 2.16.

2.4.2 Training set construction

Now that we have established a convenient way to calculate the energy of a generic dislocation configuration in a strained epi-layer, we turn to the practical construction of training sets for this task.

As we have already stated, we will consider 1+1D simulations, with the film-Si(001) substrate interface is perpendicular to y direction and dislocation lines exiting from the paper plane, as sketched in Figure 15. Periodic Boundary Conditions are applied along the x direction. In this context, formulae in Appendix C apply if the free surface is flat; otherwise, a numerical FEM solution of the mechanical equilibrium equations is required to obtain stress and strain fields.

We built a training set considering simple sinusoidal perturbations of amplitude A as a prototypical case. Keep in mind, however, that the same procedure can in principle be applied to generic boundaries. As we have to choose the computational domain size, we choose a simulation cell of 1200 nm, consistently with Ref. [66]. The perturbation wavelength has been fixed to 600 nm (see Figure 15 again). As we will be interested in the interplay between surface morphology and dislocations, different A values have

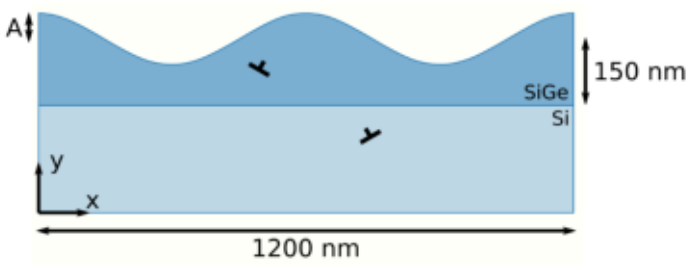

Figure 15: Sketch for 2D dislocation configuration calculations. Dislocation line $\hat{\xi}$ is exiting from the paper plane. A sinusoidal perturbation of amplitude A is superimposed to a flat, 150 nm-thick profile.

Name	Symbol	Value
Young Modulus	Y	130 GPa
Poisson ratio	ν	0.27
Burgers vector	b	0.3857 nm
Germanium concentration	c_{Ge}	0.25

Table 1: Elastic constants considered for the FEM solver used in dislocation training set generation.

been used to build various independent datasets. As a sanity check for the method, the control case of flat surface has also been included.

In practice, the construction of the training set for H_i, V_i and W_ij can be efficiently implemented as in the following points (see also the sketch of Figure 16):

- The position of 2 dislocations is chosen at random in the simulation cell. To avoid FEM ill-defined problems, an exclusion region of 10 nm from the free surface has been considered.
- Burgers vectors are assigned at random between the two possible 60° configurations available. As we will discuss, the effects of a 90° can be described by superposition.
- The Computational domain is discretized using an adaptive mesh to have a better refinement near dislocation core regions.
- The mechanical equilibrium problem is solved for the strained film *without inserting dislocations*. This way, the $\overline{\sigma}_{het}$ term of Equation 2.16 is obtained.
- The mechanical equilibrium problem is solved for the individual dislocations *setting the lattice mismatch to zero*. This way, $\overline{\sigma}_{het}$, $\overline{\sigma}_i$ and $\overline{\sigma}_j$ terms of Equation 2.16 is obtained.
- Using definitions in Equation 2.16, terms H_i, V_i and W_{ij} can be obtained through numerical integration.

In total, $\approx$ 12000 two dislocation configurations for each amplitude were collected and $\approx$ 24000 (twice as much) single body terms. Physical constants for Silicon have been used [57]. For the sake of completeness, we report their value in Table 1. Germanium concentration in the film has been fixed to 25%.

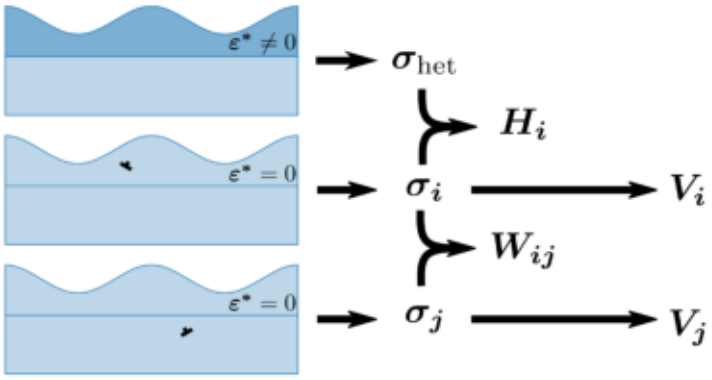

Figure 16: Schematics of the efficient training set construction procedure. The line connecting σ_j and σ_{het} to calculate H_j is not shown for clarity.

Notice that this procedure is particularly efficient, as it is required to actually solve the mechanical equilibrium problem only for meshes refined to contain only two dislocations. This is critical, as the refinement requirements near dislocation cores are one of the main reasons for the high number of elements required in the FEM solver in this class of simulations.

2.4.3 Networks definition and system symmetries

Now that an efficient energy decomposition has been established and the training set has been constructed, we need to define the NN structure. This is the first application for NN we are considering, therefore we want to keep things simple. Luckily, all functions in Equation 2.16 can be approximated by a simple, fully-connected feedforward NN. At the end of training, therefore, we will have *three* Networks for every profile shape (i.e. for every amplitude value of the sinusoid), each of which will approximate one of the terms in the energy decomposition.

For terms H_i and V_i the situation is very simple: the NN should map from $\mathbb{R}^4 \to \mathbb{R}$, as the input variables are in principle the concatenation of the 2D position vector and the 2D Burgers vector as defined in 2.11. Notice, however, that the two 60° dislocation configurations only differ in the Burgers vector y component. It is therefore more compact to use a ± 1 coding for the two different defects. H_i and V_i can therefore be simply approximated with NN functions from $\mathbb{R}^3 \to \mathbb{R}$.

W_{ij} term requires a bit more attention. First, it is possible to use the same ± 1 coding and reduce the dimensionality of the Burgers vector. Second, as observed in Section 2.4.1, $W_{ij} = W_{ji}$. It is therefore necessary to encode this exchange symmetry into the training procedure. In this case, we will use a simple data augmentation procedure, as described in Section 1.9: every training set instance for the dislocation-dislocation interaction term is duplicated and presented to the NN considering this exchange symmetry.

In terms of structure, we will keep the number of parameters to a minimum: for V_i and H_i three hidden layers with widths $(30, 20, 20)$ will be used. For W_{ij}, instead, we will turn to a more complex structure with widths $(40, 40, 40, 30)$. In all cases, the simple hyperbolic tangent activation func-

tion has been used. This set of parameters has proven to be effective, but more efficient choices could in principle be found via a a more complete (and expensive) hyperparameter search. Notice, however, that the number of weights in each NN is very contained ($\approx$ 1100 parameters for one body terms and $\approx$ 4800 for pair terms) for ML standards.

2.4.4 The loss function

At this point, we could launch a naive training procedure for elastic energy terms. As in MD, potential energy is not the whole story, however: while energetic considerations allow for the extraction of many important quantities, *forces* are essential too. Indeed, in Deep Learning potentials for molecular dynamics, energy derivatives are also fitted during the training procedure. This approach is sometimes called *Sobolev training* [84, 85], as it can be thought as the minimization of the distance between NN approximation and the true potential energy function in a suitable Sobolev space[3].

Instead of taking a detour on how Sobolev spaces enter the definition of the loss function, we will here take a more heuristic approach. Consider a NN approximating some potential energy function $E(\vec{x}) : \vec{x} \to E$. In *ab-intio* calculations, both energies E_i and forces $\vec{F}_i$ can be obtained. Since NN are differentiable functions, and their derivatives may be obtained analytically by automatic differentiation (this operation is at the core of all Deep Learning libraries), it can set an alternative minimization problem:

$$\mathcal{L}(\theta) = \frac{1}{N_{TS}} \sum_{i \in TS} [E_i - \hat{E}(\vec{x}_i|\theta)]^2 + \lambda[\vec{F}_i + \vec{\nabla}_{\vec{x}} \hat{E}(\vec{x}_i|\theta)]^2 \tag{2.17}$$

where $\hat{E}$ represent the NN approximation, θ is the set of NN parameters, TS is the training set, N_{TS} the number of training examples and λ is a hyperparameter balancing the importance of approximating energies and forces during training. The main role of λ is to address the fact that function values and derivatives may span very different ranges, depending on the chosen units. Without a normalization of the two contributions to make them comparable in size, the NN training could "collapse" and result in considering only forces or only energies. At a surface level, Equation 2.17 can be interpreted as a regularized version of standard MSE Loss training. Notice, however, that a major difference stems from the fact that predicted forces are not independent outputs of the NN. Indeed, this is a strong advantage, as there is *consistency* between energy and forces definitions for the NN model.

Since we would like to have NN potentials for dislocations which can be used both for energy estimations and Dislocation Dynamics, a similar approach should be considered also in our case. Cai's regularization proce-

3 A Sobolev space is a (function) vector space equipped with a norm which is given by a weighted sum of L^p norms of functions and their derivatives.

dure [62], ensures that energy decomposition and Peach-Koehler forces are consistent:

$$\begin{cases} -\vec{\nabla}_i V_i = (\sigma_i \cdot \vec{b}_i) \times \hat{\xi}_i \\ -\vec{\nabla}_i H_i = (\sigma_{het} \cdot \vec{b}_i) \times \hat{\xi}_i \\ -\vec{\nabla}_i W_{ij} = (\sigma_j \cdot \vec{b}_i) \times \hat{\xi}_i \end{cases} \tag{2.18}$$

where we used the shorthand notation $\vec{\nabla}_i$ to indicate derivatives with respect to dislocation i coordinates. PK contributions can be extracted from FEM calculations and are thus available for training.

The training can therefore be performed using the Sobolev loss function:

$$\mathcal{L}_S(\theta) = \frac{1}{N_{TS}} \sum_{i \in TS} \left\{ [\mathcal{E}_i - \hat{\mathcal{E}}(\vec{x_i}|\theta)]^2 + \sum_l g_l [F_l, i + \partial_l \hat{\mathcal{E}}(\vec{x_i}|\theta)]^2 \right\} \tag{2.19}$$

where $\mathcal{E}$ stands for the different energy terms and l runs on Cartesian components. Hyperparameter λ has been here divided into two different g_l coefficients. This comes from the same range considerations discussed above. Consider for example the limit example of dislocation-film interaction H_i in a flat film. In this case, the x component of the PK force vanishes. If a small perturbation is added to the flat configuration, the latter statement will no longer be true but the x contribution will be significantly smaller than the y component. To balance things out, g_l values have been chosen as the ratio between the range of F_l values in the training set and E values.

Figure 17 reports validation loss functions for both the standard MSE approach (referred to as "Value training") and Sobolev training as a function of the number of training epochs. The specific case chosen here is the dislocation-film interaction term H_i in the case of $A = 60$nm. The training was performed using Adam optimizer [40] with a learning rate of 10^{-5} and was stopped when the relative reduction in validation loss was smaller than 10^{-6} in 1000 epochs. As can be observed, loss 2.19 leads to a lower loss model. In both cases, there is no sign of overfitting, as validation loss decreases during the whole training (up to oscillations related to stochastic minimization and momentum). This is to be expected, as we are in a regime in which the number of parameters is much smaller than the number of examples.

$\mathcal{L}$ values comparison may be misleading, as possible trends and biases in the learned NN cannot be fully captured by a single number. For this reason, during training of scalar functions (or small dimensional vector-valued functions), it is customary to inspect the so-called *regression plot*. This is obtained by plotting the quantity predicted by the NN for validation (or test) examples versus the actual quantities. Deviations from the bisector line (here reported in orange), allow for a visual assessment of the generalization capabilities of the NN. Root Mean Squared Error(RMSE) values are also reported as a second figure of merit as insets in the regression plots of figure 18(a,b). It is clear that Sobolev training not only leads to a much more accurate prediction of forces (which is to be expected, as they are explicitly passed to the NN during training) but also has beneficial effects on the accuracy of

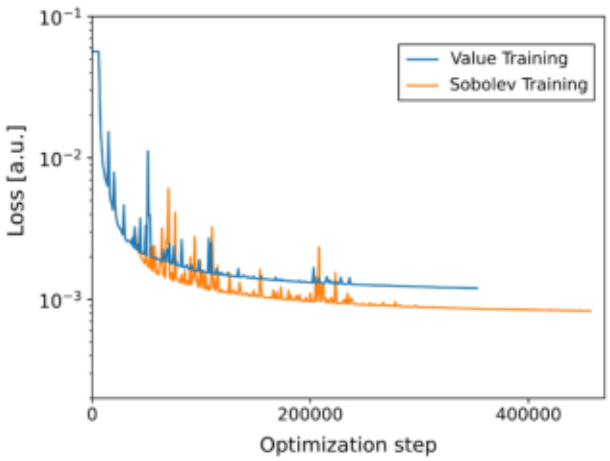

Figure 17: Training loss for Value and Sobolev training comparison

energy predictions, consistent with other studies [84, 85]. Indeed, RMSE for energies reduces by almost a factor of 3.

Additional tests can be found in Ref. [55].

2.5 FEM WITHOUT FEM

Once a NN model is trained and its quality has been assessed, we can use it to perform simulations which would be unfeasible using traditional methods. As a proof of concept, we report here two different applications: low energy dislocation configurations search and simple Dislocation dynamics simulations.

2.5.1 Minimum Energy configurations

As is made evident from the discussion on FEM costs, the search for global (or even local) minima for dislocations in strained films is a computationally intensive task, as a huge number of FEM calculations are potentially required.

While in complex systems such as epi-layers containing multiple defects, the probabilities of observing equilibrium configurations may seem slim, in the context of semiconductor heterostructures such results have interesting connections with experimental images. Indeed, it has been shown experimentally that under specific conditions dislocation distributions resembling equilibrium ones can be observed [86, 87]. Knowing the energy cost (or gain) for introducing dislocations is also useful to understand thermodynamic driving forces. Such description has proven a valuable tool for simulating dislocations in thin films [57, 76, 72], islands [88, 89, 73, 76] and vertical heterostructures [90, 91, 92, 93]. Notice that in general, even if real systems do not present equilibrium configurations, thermodynamic information (as opposed to more complete kinetic pathways) still represents the starting point for the study of dislocation configurations in non-trivial systems.

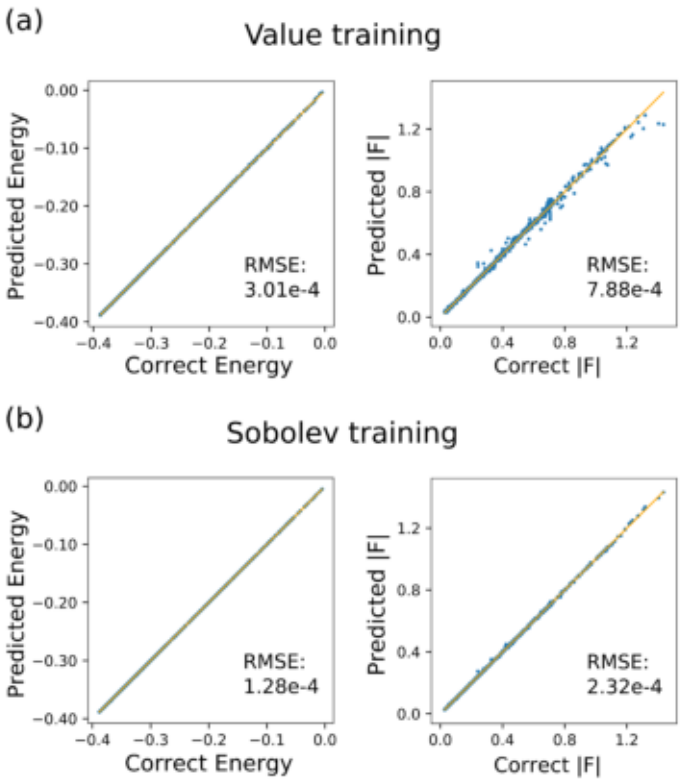

Figure 18: Regression plot comparison for Value (a) and Sobolev training (b) for the dislocation-heteroepitaxial field interaction term H. RMSE on predicted values is also reported ad inset.

In particular, here we will use a simple Monte Carlo scheme derived from standard Metropolis algorithm [94]. First, the number of dislocations N and the amplitude of the profile perturbation A is chosen and an initial dislocation distribution is extracted at random. The method proceeds to randomly extract new configurations by moving one dislocation in the system to a new, random position. Total elastic energy is calculated using decomposition 2.16 and trained NN approximations. The new configuration is retained with probability proportional to $\exp \frac{-\Delta E}{k_B T}$, where ΔE represents the variation of the elastic energy, k_B is Boltzmann constant and T is a temperature parameter.

To exploit the temperature role in surpassing energy barriers and exiting shallow minima, but still preferentially sampling low-energy configuration, a simple simulated annealing protocol has been implemented: T was initially set to 0 K, then progressively increased to 1000 K and was brought back to 0 K again. Energy minimization accepting only energy-decreasing moves had finally been performed. While this approach is neither elegant nor particularly sophisticated, in our case led to good results. Notice that full FEM searches using this approach would require thousands (possibly millions) of energy evaluations. In this application, we estimated that the same precision used for the training set would have required approximately 2 years (of course this number can be decreased if HPC infrastructures are used). Searches conducted using NN approximations, however, required approximately a few minutes, providing a dramatic computational speed-up.

Before launching configuration searches, we have yet to define dislocation behavior in terms of their Burgers vector. As discussed in Section 2.2.5, three different Burgers vector choices are possible in the system, corresponding to 90° sessile dislocations and the two families of 60° mobile dislocations. In the training set, however, only the latter are explicitly present. This can be addressed with a straightforward procedure: when complementary dislocations get closer than 5 nm, they undergo a reaction and form the corresponding 90° defect. Notice that the self-energy of a 90° dislocation is not merely the sum of the self-energies of 60° dislocation. This extra contribution can however be accounted for by the W_{ij} term, calculated with the two dislocations at the same point. Interactions between 90° and other dislocations, instead, can be calculated directly by superposition.

Notice that the Monte Carlo algorithm described does not comply with the glide motion one would expect from dislocations in epi-layers, especially at low temperatures. This is done on purpose, as the main goal here is to search minimum energy configurations and not to reproduce real Dislocation Dynamics. In this spirit, the sessile character of 90° dislocations is also removed. For the same reason, it is possible for 90° dislocations to actually decompose in the two complementary 60° dislocations.

Figure 19(a,b) reports minimum energy configurations found for flat morphology containing 8 dislocations. To observe differences, we both placed identical 60° dislocations (panel a) and alternating Burgers vectors (panel b). As can be observed, in both situations dislocations place at the SiGe/Si(001) interface, as expected from analytical theory (see [72] for a direct comparison) and some experimental observations (see [86, 95], where an ad-hoc annealing procedure was used to promote 90° dislocation formation through reactions).

The same analysis has been performed on non-flat surfaces, where no analytical solutions are available. Results are reported in Figure 19(c) for the $A = 60$ nm case. Here reactions are clearly inhibited and dislocations are placed underneath valleys in the free surface profile, aligning their stress field to the over-compression caused by the non-flat morphology. The side on which they place depends therefore directly on the Burgers vector. Results are consistent with previous studies for islands, such as those reported in References [96, 97, 88]. As a further check of the quality of NN approximation, FEM one-shot calculations on such minimum energy configurations have been performed. The relative deviation in energy amounts to approximately 2‰.

Given that the results are physically sound and the low cost of this kind of Monte Carlo search, an extensive study of the interplay between dislocation positioning and surface morphology has been performed. A total of 500 searches for each non-flat profile were performed and dislocation densities as a function of x have been calculated. In order to obtain a continuous function, dislocation positions have been smeared using Gaussian kernels. Figure 20 reported dislocation densities in red and blue for 60° dislocations, depending on the y component of the Burgers vector, and in green for 90° dislocations. The trend is clear: if the amplitude of the free surface perturbation is high, dislocations are more and more localized at the sides of sinusoid minima, testifying that the configuration shown in Fig. 19(c) is rep-

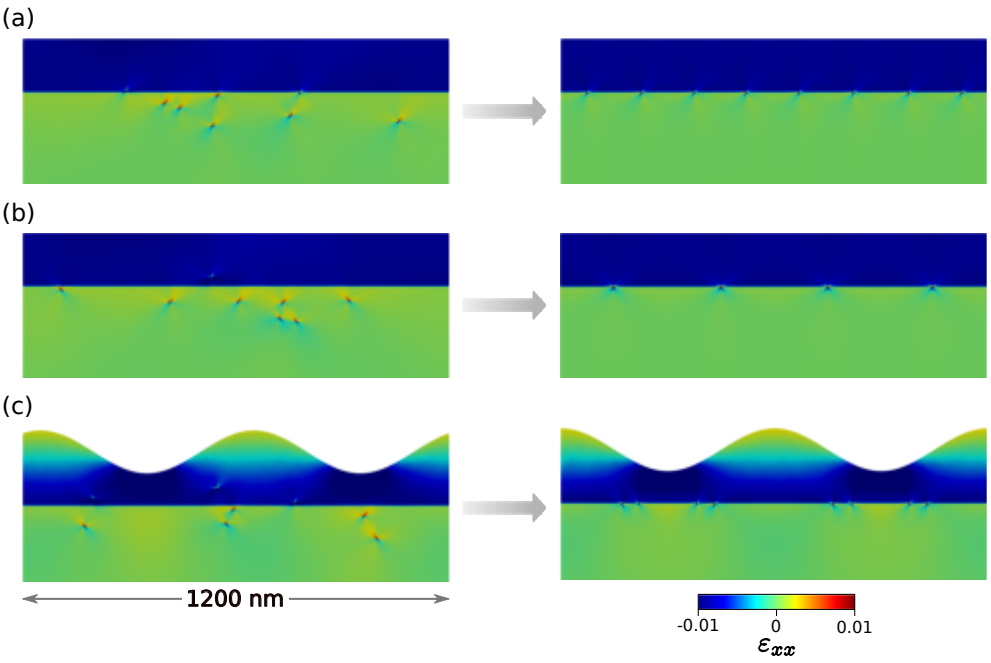

Figure 19: Minimum energy configuration results. In the case of flat surface, both when the system is initialized with identical dislocations (a) or with complementary 60° defects (b), minimum energy configuration yields an equispaced array. In the latter case, dislocation reaction is present and 90° dislocation form. This is no longer true in the case of large profile undulations (c).

resentative. On the other hand, for moderate perturbations ($A = 15$ nm and $A = 7.5$ nm), dislocation confinement is weaker and there is no longer total elimination of reactions. As expected, when the free surface profile is flat, dislocations are spread out in the simulation cell. Local minima in which no total reaction has been achieved contribute to the small density of 60° dislocations.

2.5.2 Beyond the training set: large systems simulations

Simulations performed in the previous section were conducted on a computational domain with the same size as the training one. The trained NN functions, however, can be used to evaluate energies of dislocation configurations on possibly much larger domains, provided that surface profiles have the same shape and periodicity. We will now discuss under which conditions such simulations can be performed and provide an analysis of dislocation low energy configurations in a computational domain that is 4 times larger than the training set one.

Dislocation-film interaction terms H_i and dislocation self terms V_i are invariant under translation of periodicity vector of simulation cells $\vec{R} = [1200\text{nm}, 0]^T$. The energy of an isolated dislocation in an extended cell can therefore be obtained by NN approximation by simply re-mapping its x coordinate in the original training set cell. This does not introduce additional errors with respect to NN approximation ones.

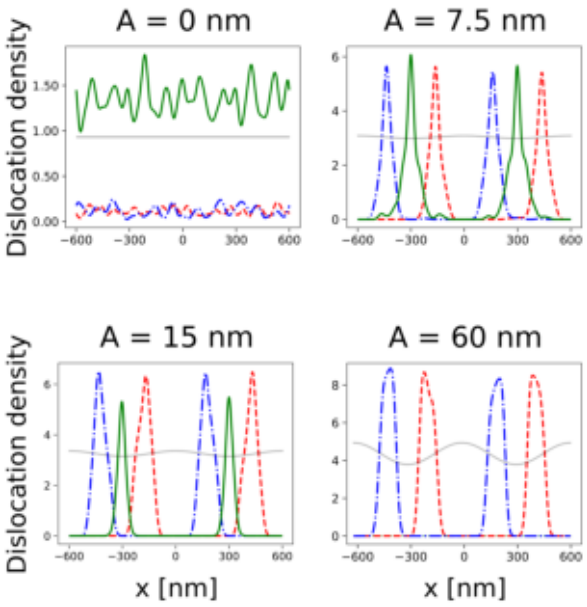

Figure 20: Dislocation densities evaluated by extensive minimum energy configuration searches. Dislocation reaction is favored for the flat configuration, while unreacted dislocations provide a better strain relaxation for strong perturbations.

The dislocation-dislocation W_{ij} term should be treated with more care. As there are no dislocations more than 600 nm apart along x in the training set, the NN approximation would yield extrapolation errors if energy evaluation is attempted. A simple solution would be to disregard such interactions. Dislocation stress fields (hence interaction energy terms), however, are slowly decaying [98, 67]. It is therefore crucial to have an estimation of the additional approximation error that would be introduced. In the non-flat surface case, this is computationally expensive to do, as FEM calculation costs increase with computational domain size. For flat films, however, analytical expressions in Appendix C are available. It turns out that for a periodicity of 1200 nm, the error introduced amounts to approximately 1% of the interaction energy (see Supplemental material of Ref. [55]).

Having justified the cutoff scheme, low-energy configuration searches can be performed using the same Monte Carlo approach of the previous Section. The minimum energy configuration found in 300 independent runs is shown in Figure 21(a). As expected, dislocations place below valleys in the free surface, as in the 1200 nm case. The relative difference with a one-shot FEM calculation of the elastic energy amounts to approximately 3‰. The elastic energy of the system is plotted as a function of the variance of the number of dislocations beneath each valley in the sinusoidal profile function in Figure 21(b) (error bars represent the error with FEM calculation of configuration in panel(a)). This shows the presence of inverse proportionality between the elastic energy of the system and how evenly distributed are dislocations, as expected from symmetry considerations.

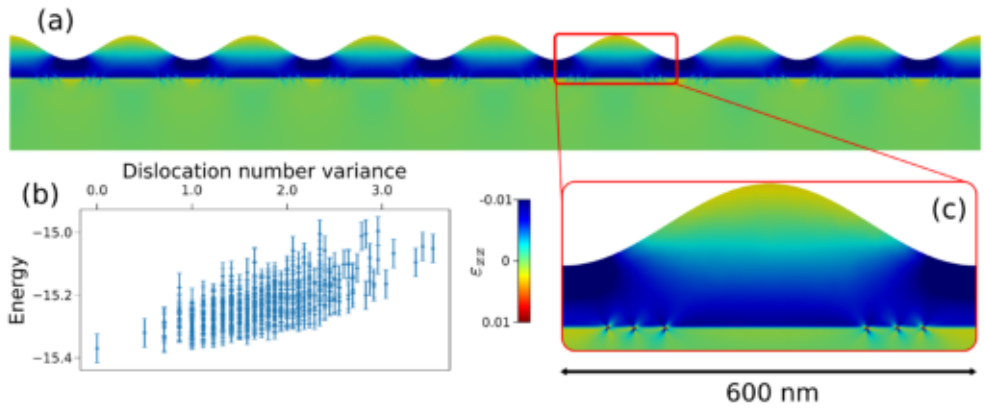

Figure 21: Minimum energy configuration found for a 4800 nm domain size, extending over the training set 1200 nm computational cell (a). The energy of a dislocation configuration in the same cell as a function of the variance of dislocations below each free surface minima (b). (c) reports a detail of panel (a).

2.5.3 Dislocation dynamics

Equilibrium dislocation configurations are interesting on their own, but they are not the whole story. The real motion of dislocations can also be simulated if forces acting on each defect are known. There are multiple approaches to implement Dislocation motion in simulations [54]. For what we are concerned about, we will follow a Discrete Dislocation Dynamics approach, in which dislocations are moved along the PK force projected on their glide plane. In 2D, this reduces to a simple dynamical system of point-like objects.

The final configuration of a simple simulation is shown in Figure 22 (glide planes are highlighted with dashed lines): complementary 60° dislocations were placed high in the protruding features of the profile and glide pushed them to the SiGe/Si(001) interface. Notice, that the results strongly depend on the Burgers vector of dislocations. On the left, glide planes lead defects close to their ideal position. On the right, instead, Burgers vectors are exchanged and dislocations merge in a 90° dislocation. As the glide plane for this defect is parallel to the x axis, the dislocation is now pinned. The ideal position of a 90° dislocation is directly below one of the minima in the free surface, as can be predicted by symmetry considerations of the stress field lobes. This, however, cannot be reached, as PK force has no components along x.

2.6 WHERE SHOULD WE GO FROM HERE AND WHAT IS MISSING

In this Chapter, we have shown the first application of ML methods in accelerating computational Materials Science approaches of this Thesis. Even if very simple NNs were used, results are promising and Deep Learning interaction potentials could provide an interesting tool to study dislocation

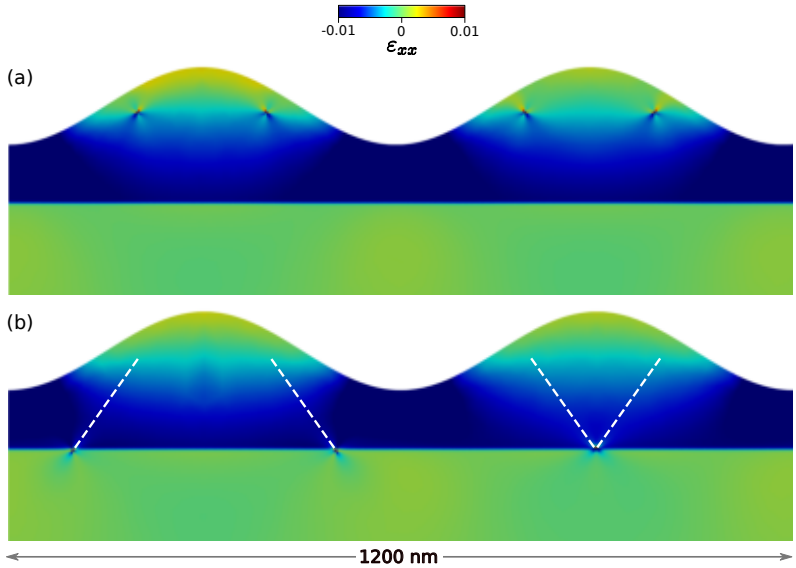

Figure 22: Simple 2D dislocation dynamics simulation performed using the trained potentials. Depending on the different orientations of the glide planes (highlighted as dashed lines) dislocation reaction may be forced or prevented.

behavior in nanostructures, where a high level of precision on dislocation individual positions is required.

Of course, there are still a lot of pieces missing. Maybe the most limiting factor is that the current approach only works in 2D. While this allows one to understand many critical behaviors, real heterostructures cannot be realistically approximated as infinite in one direction except for specific geometries. In principle, extension to a full 3D description is straightforward: energy decomposition stems from linear elasticity and can be readily applied, for example. Still, the computational costs of 3D systems make it much more expensive to build datasets. Additionally, dislocation lines are generally *not straight*, making it complex to fully describe a dislocation configuration. Some ideas may come from learning *dislocation segments* energies and interactions (see [49] for analytical expressions in bulk, for example) or describing dislocation arcs in terms of some basis set (e.g. Fourier components of a parametric curve). To our knowledge, an elegant and efficient solution to this problem is still missing.

Another main downside of the approach as was presented in this Chapter is that we need different models for H, V, and W *for each amplitude* of the free surface perturbation A. An actual breakthrough would be to suitably include the actual surface morphology within the NN inputs. This would allow for fast simulations for arbitrary profiles, providing true general-purpose dislocation potentials. Even better, such an approach could be merged with the one presented in Chapter 3, where surface morphology evolution itself is discussed. This would enable a unified framework that tackles morphology evolution *in the presence* of dislocations.

Implementation of full anisotropic elasticity and elastic constant dependence on Ge concentration, on the other hand, are straightforward additions and just amount to the construction of dedicated training sets.

3 MORPHOLOGICAL EVOLUTION BY CONVOLUTIONAL NN

In the previous Chapter, we analyzed how NN approaches can significantly speed up simulations involving dislocations in a strained film. The free surface shape, however, was considered fixed. In real materials, of course, this is not the case. In the present Chapter, we will therefore turn to the complementary problem of dislocation-free strained film morphological evolution. In partial continuity with the previous Chapter, applications will be shown in the case of pure Ge on top of a Si(001) film. As it will be described in detail, the main bottleneck for this class of simulations is again related to the necessity of solving the elastic equilibrium problem. The presented approach, however, is general and may be applied in the future to a broad class of systems.

The Chapter is organized as follows. Section 3.1 presents a review of Mullins' model [99] describing the mathematical framework in which free surface evolution will be discussed in this Chapter.

Section 3.2 outlines the main features of Convolutional Neural Networks. In particular, we will show how this architecture is not a mere application of a popular DL framework, but symmetry and physical considerations call for this specialized NN architecture in the application to the problem at hand.

The following two Sections report the results of applying Convolutional Neural Network to predict the elastic contribution to the chemical potential, bypassing explicit numerical solutions. Section 3.3 applies CNN to approximate the chemical potential obtained with a computationally cheap, small-slope approximation. This simplification allows for the generation of a very large dataset, which in turn makes an in-depth analysis of the model accuracy possible. Once the NN approach is confirmed to be reliable, morphological evolution simulations are conducted. Section 3.4 transfers the same procedure to FEM data, yielding dramatic computation speedups.

Conclusions and perspectives are discussed in Section 3.5.

Results of this Chapter, together with more in-depth analyses, are collected in a paper in preparation at the time of writing. The datasets and additional simulations can be already found at [100].

3.1 MORPHOLOGICAL EVOLUTION AND SHARP INTERFACE MODELS

Modeling surface morphology and its evolution is a classical topic in Materials Science simulations [99, 101, 102, 103]. Depending on the time and spatial scales involved, there are plenty of different approaches that can describe processes at surfaces and interfaces. Limiting factors and advantages are the same as discussed in dislocation modeling: while atomistic approaches can provide details on phenomena regarding electronic structure (*ab-initio* DFT

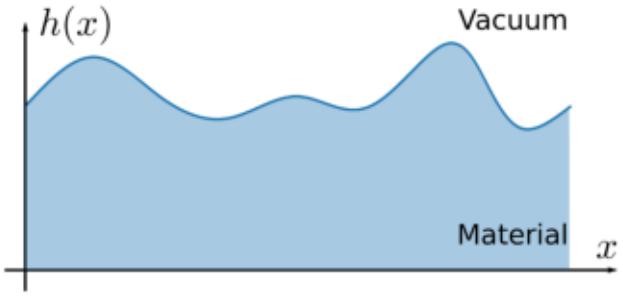

Figure 23: Sketch representing the free surface of a material using Mullins sharp interface approach through a profile function $h(x)$.

calculations [104, 105]) and surface configurations (e.g. molecular dynamics [106, 107] and Kinetic Monte Carlo approaches [108, 109]), continuum approaches are at the moment the only method which is capable of tackling "human" (seconds or longer) timescales.

3.1.1 Thermodynamic potentials

One of the most successful approaches traces back to the pioneering work from Mullins in 1957 [99]. In his approach, the free surface of a solid is described by a profile function $h(x, z)$. This assumption, while quite general, does not cover all possible morphologies as it excludes non-convex shapes. The treatment can be generalized to parametric surfaces [103], however. In the following, the surface will be assumed as uniform in the z direction (which is set perpendicular to the paper plane), so that observing the morphological evolution of a slice would be enough to fully characterize the system (see sketch in Figure 23). This picture is sometimes called a 1+1D approach, as the surface morphology can change only in one direction and the profile is described by a function of a single variable $h(x, z) \to h(x)$. This setting is the same we used in 2D simulations of dislocations in Chapter 2. In this sense, the treatment we will develop here may be regarded as the complementary problem of dislocation-free, stressed films which, however, evolve in time.

The free energy[1] F_γ associated with a free surface can be defined in general through the following integral:

$$F_\gamma = \int \gamma(\hat{n}) dS \tag{3.1}$$

where dS is the surface element and $\gamma(\hat{n})$ is the *surface energy* of the material, corresponding to the (free) energy required to increase the free surface of the body by a unit, i.e. $\gamma = \delta F / \delta S$. An analogous treatment may be derived for interfaces between two different phases of the same material or interfaces between two substances. For crystalline solids, surface energy depends on the exposed orientation, as the number of dangling bonds and entropic contributions will depend on the crystallographic facets. This is in-

1 As we are dealing with solids, the difference between Gibbs and Helmholtz free energies is usually negligible as the pressure-volume term is usually small at ambient or regular conditions.

dicated by the dependence on $\hat{n}$, the free surface normal. In this Thesis, we will work on the assumption of *isotropic* surface energies, such as those of poly-crystals and we will therefore drop the explicit dependence on $\hat{n}$. This simplifying assumption is mainly motivated by mathematical and numerical convenience: anisotropic simulations require more care and, in general, lead to more expensive datasets. Extensions in this direction are one of the future perspectives for future works. As we are concerned with a 1+1D picture, the surface integral may be reduced to a line one:

$$F_\gamma = \int \gamma dl = \int \gamma \sqrt{1 + h'(x)^2} dx \tag{3.2}$$

The generalized thermodynamic potential conjugated to a thermodynamic quantity X can be defined as $\delta F/\delta X$. Using this definition and standard thermodynamics we can derive the generalized chemical potential associated with the presence of the free surface as the functional derivative $\delta F_\gamma/\delta h$. By direct application of Euler-Lagrange equations [110]:

$$\mu_h(x) = \frac{\delta F_\gamma}{\delta h} = -\frac{d}{dx}\frac{\partial}{\partial h'}\gamma\sqrt{1 + h'(x)^2} = -\gamma\frac{h''(x)}{(1 + h'(x)^2)^{\frac{3}{2}}} \tag{3.3}$$

The physical meaning of this term can be easily understood considering the *surface curvature* κ. Given a generic profile function, this quantity involves regular derivatives of $h(x)$ and can be simply derived from standard calculus:

$$\kappa(x) = -\frac{h''(x)}{(1 + h'(x)^2)^{\frac{3}{2}}} \tag{3.4}$$

With this convention, points with positive curvature are locally convex, and points with negative curvature are locally concave. The direct proportionality between surface curvature and chemical potential is consistent with traditional results, such as the Gibbs-Thompson relation [111].

As long as no other energy contributions are present, the equilibrium configurations of the free surface of an isotropic body, being the total volume occupied V fixed, may be easily obtained considering the constrained minimization of the free energy F_γ. We can therefore consider the functional $A(h, h')$:

$$A(h, h') = F_\gamma + \lambda V = \int \gamma\sqrt{1 + h'(x)^2} dx + \lambda \int h(x) dx \tag{3.5}$$

where λ is a Lagrange multiplier. The second integral in the above equation represents the total volume occupied by the material in the 1+1D picture[2]. Using Euler-Lagrange equations again, we have

$$\frac{\delta A}{\delta h} = \kappa\gamma + \lambda = 0 \quad \Longrightarrow \quad \kappa = -\frac{\lambda}{\gamma} \tag{3.6}$$

2 The same expression, however, may be obtained considering the total free energy as the sum of the surface contribution and the bulk free energy. In that case, the role of λ is that of a bulk energy density.

In other words, the curvature of the solid free surface should be constant, i.e. the material should have a circular perimeter in 2D. This is consistent with the intuitive idea that a 3D isotropic solid should have a spherical shape, as the sphere is the solid that minimizes the surface-volume ratio. This is not, however, the only possible shape encompassed by condition 3.6. Setting λ to zero, we obtain that a zero-curvature surface is also an equilibrium configuration. A flat free surface is not a pathological case: if we consider PBCs, which is very convenient from a computational point of view, and only consider simple profile functions $h(x)$, this is the only equilibrium configuration which can be obtained[3].

Naturally, this discussion is valid only if free surface energy is the only contribution. In general, other terms in F will be present. One example is the elastic energy stored in strained films which will be discussed in Section 3.1.3.

3.1.2 Evolution by surface diffusion

The generalized chemical potential in Equation 3.3 can be used to derive the driving force for different non-equilibrium processes. In his original work, Mullins defines two main regimes, in which surface evolution is limited either by attachment-detachment of atoms or by surface diffusion [99]. In this Thesis, we will mainly deal with the latter case and refer the interested reader to Appendix D for a discussion of the first case. Of course, these represent limiting cases and more general situations may be derived as combinations of these regimes or by addition of suitable terms.

The assumptions for the model are that atoms on the surface diffuse, there are no attachment-detachment processes and bulk diffusion is negligible. The chemical potential for a particle on the surface may be obtained from μ_h as

$$\mu = \mu_h V_a = \kappa \gamma V_a \tag{3.7}$$

where V_a is the atomic volume, acting as a conversion factor between generalized chemical potential and the proper $\delta F/\delta n$, being n the number of particles.

Using Onsager linear laws [112], we may derive the atomic diffusion flux $\vec{J}$ on the surface, which is proportional to minus the gradient of the chemical potential, i.e. atoms diffuse from regions of high to regions of low chemical potential, consistently with μ physical meaning:

$$\vec{J} = -D_s \vec{\nabla}_s \mu = -D V_a \vec{\nabla}_s \mu_h \tag{3.8}$$

where D_s is a *diffusion constant* and the subscript s for the gradient operator indicates that derivatives should be taken along the free surface manifold. In principle, D_s may not be a simple scalar but depends on orientation for

3 This simple analysis may be generalized to closed, possibly non-convex bodies using parametric surfaces, yielding the circular solution. Generalizing to full 2D parametric surfaces yields indeed a sphere.

anisotropic materials. In the following, however, we will assume atoms on the surface have isotropic mobility.

The free surface normal movement should be proportional to the accumulation or depletion of diffusing particles. In mathematical terms, this quantity may be expressed as the surface divergence of $\vec{J}$, hence the normal velocity of the free surface will be:

$$v_{\hat{n}} = -D_s \nabla_s^2 \mu \tag{3.9}$$

where ∇_s^2 represents the Laplacian on the free surface and we absorbed all constants (e.g. V_a) in D_s. Turning to the 1+1D picture, we have that divergence turns to second order spatial derivative along the line described by h, $\nabla_s^2 \to \partial/\partial l$. Noticing that

$$\begin{aligned} \frac{\partial}{\partial l} &= \frac{1}{\sqrt{1+h'^2}} \frac{\partial}{\partial x} \\ v_{\hat{n}} &= \frac{1}{\sqrt{1+h'(x)^2}} \frac{\partial h}{\partial t} \end{aligned} \tag{3.10}$$

we obtain the non-linear PDE for $h(x,t)$:

$$\frac{\partial h}{\partial t} = -D_s \gamma \frac{\partial}{\partial x} \left[\frac{1}{\sqrt{1+h'^2}} \frac{\partial}{\partial x} \left(\frac{h''}{(1+h'^2)^{\frac{3}{2}}} \right) \right] \tag{3.11}$$

The equation conserves the average value of $h(x)$, as it should be expected from diffusive dynamics [4].

This can be reduced to a linear PDE in the small slope limit $h' \to 0$:

$$\frac{\partial h}{\partial t} = -D_s \gamma h'''' \tag{3.12}$$

3.1.3 Elastic contributions

In the case of a strained film, the driving force for morphological evolution will no longer be the simple expression 3.3. In principle, adding elastic contributions is as easy as operating the substitution

$$\mu_\gamma \to \mu_{tot} = \mu_\gamma + \mu_\varepsilon \tag{3.13}$$

in Equations 3.9, where μ_γ is the $\kappa\gamma$ curvature term and μ_ε is the *elastic chemical potential*.

Deriving an expression for μ_ε in the 1+1D model is straightforward. From linear elasticity theory, the deformation-free energy density of a strained

4 A simple proof is the following: consider that the Fourier transform of $\partial h/\partial t$ vanishes at the origin for all times, as $\partial/\partial x \to -iq$ upon transformation. Remembering that the value of the Fourier transform at the origin yields the mean value of the original function, this means that the mean value of h is constant, as should be expected for a conserved quantity.

body may be calculated if the stress and strain films are known. The free energy associated with elastic terms is therefore

$$F_\varepsilon = \int \int_{-\infty}^{h(x)} \frac{1}{2} \overline{\sigma}(x,y) : \overline{\varepsilon}(x,y) dy dx \tag{3.14}$$

where the : symbol indicates tensor product as in Appendix B. using Euler-Lagrange equations, this simply yields

$$\frac{\delta F_\varepsilon}{\delta h} = \frac{1}{2} \overline{\sigma}(x,h(x)) : \overline{\varepsilon}(x,h(x)) \tag{3.15}$$

The total chemical potential, if only curvature and strain effects are present, then reads:

$$\mu_{tot} = \mu_\gamma + \mu_\varepsilon = V_a \left(\kappa\gamma + \frac{1}{2} \overline{\sigma} : \overline{\varepsilon} \right) \tag{3.16}$$

where stress and strain fields are calculated at the material surface and V_a has the usual meaning of atomic volume. Evolution equations for strained films may therefore be obtained both in attachment-detachment and surface diffusion regime by simple insertion of Equation 3.16 in Equations D.5 and 3.9 respectively.

This, however, is easier said than done. In general, assuming that elastic processes happen on timescales much shorter than atomic diffusion and condensation/evaporation, elastic fields can be obtained by solving the mechanical equilibrium problem, which is equivalent to setting the elastic relaxation instantaneous. This in turn calls for a numerical solution of stress/strain fields, if the system configuration is not the trivial flat surface. As we are interested in the case of strain originating from lattice mismatch, the elastic energy term may be obtained numerically by FEM using the same eigenstrain approach used in Chapter 2. This introduces the main computational bottleneck in this class of simulations, even in the dislocation-free case we are considering: while the requirement of a high density of elements near dislocation cores is no longer required, time integration procedures require elastic energy calculations for thousands (if not millions) of iterations when tackling real materials behavior.

3.1.4 Small slope approximations

As is common in physics, analytical expressions may be obtained in some linear regimes. In particular, in small slope regimes strain fields for a biaxially strained, coherent film can be obtained using elastic Green functions for a half space [48, 113]. Calculations are rather lengthy and we will not derive them here. However, we will exploit the final expression in the following to construct a training set, hence we report it here for completeness. Under normal solid assumptions (i.e. homogeneous and isotropic, Appendix B), the xx component of the strain field reads:

$$\varepsilon_{xx}(x) = -\frac{Y\varepsilon^*}{1-\nu^2} \int g_{xx}(x-x')h(x)dx' = -2\varepsilon^* \int \frac{h(x)}{(x-x')^2} dx' \tag{3.17}$$

where g_{xx} is the Green function relating a force density to the strain field directly and ε^* is the eigenstrain term as in Appendix B and defined through lattice mismatch as in the previous Chapter. The only other non-vanishing strain component is $\varepsilon_{yy}(x) = (\varepsilon^* - \nu\varepsilon_{xx})/(1-\nu)$, fully determining the elastic chemical potential term:

$$\mu_\varepsilon(x) = V_a \frac{Y}{2(1-\nu^2)}(\varepsilon_{xx}(x) - \varepsilon^*)^2 \,. \tag{3.18}$$

While small slope limits have only limited predictive power, an important behavior may be derived. Consider a free surface which is constituted by a flat profile perturbed by a sinusoidal term with wave number $q = 2\pi/\lambda$. If the perturbation is small, linearization of the governing equations is justified. The amplitude $A(t)$ of such profile can be obtained through linear stability analysis and grows exponentially as [113]

$$\begin{cases} A(t) = A(0)\exp\left[-K\gamma V_a q(q-q_c)t\right] \\ A(t) = A(0)\exp\left[-D_s\gamma V_a q^3(q-q_c)t\right] \end{cases} \tag{3.19}$$

The two equations are for attachment-detachment and surface diffusion-driven dynamics respectively. The quantity $q_c = \frac{2Y}{1-\nu^2}(\varepsilon^*)^2$ is the critical wavenumber. The prefactor of time in the exponential is the *amplification factor* a:

$$\begin{cases} a(q) = -K\gamma V_a q(q-q_c) \\ a(q) = -D_s\gamma V_a q^3(q-q_c) \end{cases} \tag{3.20}$$

for the attachment-detachment and the surface diffusion limited cases respectively.

The physical content of Equation 3.19 may be expressed compactly as follows: whenever a strained film undergoes evolution (either by attachment-detachment or surface diffusion dynamics), it is actually energetically favorable to corrugate its morphology. While this increases the exposed surface, yielding a higher cost in terms of $\kappa\gamma$ term, the same free surface allows the strained film to laterally expand, a process called *elastic relaxation* of the film.

Of course, not all corrugations are created equal. Short wavelength perturbations with $q > q_c$ decay exponentially fast in time, while long wavelength terms with $q < q_c$ grow in time. Of course, this analysis is valid only as long as the amplitude of the perturbation is small. This is in striking contrast with the non-strained case, in which all perturbations decay exponentially, as can be easily proven by inserting $h(x) = A\cos qx$ in Equation D.7 and 3.12. As a reference value, the critical wavelength for pure Ge on Si(001) in our simulations will $\lambda_c \approx 17$ nm (as can be obtained by inserting constants provided in Table 2), which sets the typical size of features expected the system which will be considered in this Chapter. This phenomenon is called Asaro-Tiller-Grinfeld (ATG) instability from the name of their discoverers [114, 115, 116].

3.1.5 Wetting and Ge strained films

In the following, we will deal again with the germanium-silicon system, albeit pure germanium will be considered instead of a SiGe alloy. In this context, there is an additional important contribution to the chemical potential which has to be considered, which is related to Ge tendency to form a layer spreading on a Si substrate, called *wetting layer*. The physical motivation for this phenomenon is related to the fact that Silicon has a higher surface energy density than Germanium [117, 57]. Ge-Si interactions are therefore favored with respect to Ge-Ge or Si-Si ones. As the film grows thicker, however, this effect will be less important, as Silicon dangling bonds become progressively saturated. As elastic effects are present, at some point the ATG instability (or some analogous mechanism) will lead to the formation of separated, island-like features. This morphological evolution pathway is referred to as *Stanski-Krastanov growth* [57].

This thickness-dependent behavior in surface energy may be inserted in the model outlined in previous sections through a corresponding height-dependent surface energy density $\gamma = \gamma(h)$. The exponential interpolation proposed in Ref. [117] between Si and Ge values, will be used in the present work as an effective approximation:

$$\gamma(h) = \gamma_f + (\gamma_s - \gamma_f)\exp(-h/d) \tag{3.21}$$

where γ_f is the surface energy constant of the film and γ_s represents the surface energy constant of the substrate. In the specific case at hand, $\gamma_f = \gamma_{Ge}$ and $\gamma_s = \gamma_{Si}$. If this expression is inserted back in the surface free energy expression 3.2 and Euler-Lagrange equations are used, we arrive at the following expression for the total chemical potential:

$$\mu_{tot} = \mu_\gamma + \mu_\varepsilon + \mu_w = V_a\left(\gamma(h)\kappa + \frac{1}{2}\overline{\sigma} : \overline{\varepsilon} + \frac{1}{\sqrt{1+h'^2}}\frac{\gamma_f - \gamma_s}{d}e^{-\frac{h}{d}}\right) \tag{3.22}$$

Notice that almost all quantities can be obtained by some simple finite difference scheme or through analytical definition even out of the non-small slope limit. The only exception is the computationally heavy elastic energy term.

3.2 CONVOLUTIONAL NEURAL NETWORKS

As we discussed in Section 3.1, the main problem in tackling realistic strained film behavior is the solution of mechanical equilibrium equations from linear elasticity, which yields one of the driving forces to material evolution. From a ML perspective, this problem is in many aspects analogous to the one presented in Chapter 2: we need to find a fast and reliable approximation to map the system configuration $h(x)$ to a corresponding energy term $\mu_\varepsilon(x)$. As in all numerical methods, we can discretize x, yielding a set of C collocation points. One may therefore be tempted to replicate the same procedure of dislocation potentials and define a feedforward fully-connected NN $\hat{\mu}_\varepsilon : \mathbb{R}^C \to \mathbb{R}^C$. This naive approach, however, has many drawbacks.

First, once the NN is trained, we are constrained on the number of collocation points we can handle: should we want to simulate a system that is twice as large, there is no trivial way, at least for now, to exploit the trained $\hat{\mu}_\varepsilon$ to obtain the corresponding map operating on the new domain $\mathbb{R}^{2C} \to \mathbb{R}^{2C}$. Indeed, this is the same problem we discussed in Section 2.3. There, the solution turned out to be a decomposition of the total energy into local contributions. Here, instead, we will show that a specialized NN architecture may be used.

Second, a fully connected architecture is probably wasting a lot of parameters. This can be easily seen considering that the mapping we are interested in is *equivariant* under spatial translations [22]. A function f is said to be equivariant to some symmetry operation T, if the application of T to its inputs translates to the the application of the same operation to the output. Mathematically $f(T(x)) = T(f(x))$, i.e. the function and the symmetry operation commute. Indeed, if we shift horizontally the profile function, we expect that (at least in an infinite system or if PBCs are used) the corresponding chemical potential should be identical up to the same translation operation. This symmetry must therefore correspond to a constraint to parameter values.

This is not only an advantage from a computer memory and computational efficiency point of view, however. Symmetry compliance of NN architectures usually allows also for an increase in the generalization capabilities, as discussed in Section 1.9. As it turns out, a specific NN architecture was invented in the late 80s, which is by construction equivariant with respect to spatial translations: Convolutional NN (in short CNN) [118, 119].

3.2.1 From fully-connected to convolutional Networks

Convolutions are usually directly introduced in ML courses for images, as one of the main fields in which CNNs are used in modern machine learning is in computer vision tasks. In this Section, however, we want to exploit the simpler context of mapping vectors to vectors to draw a more deep connection with standard fully connected structures. Indeed, if some specific symmetry assumptions are imposed on a fully-connected NN, we will see that convolutional structures emerge naturally. The reader who is familiar with CNNs, however, is free to skip to section 3.3, where applications will be discussed.

We will now analyze the problem from a more formal perspective. If the profile function $h(x)$ is discretized on a uniform grid of C collocation points, then it can be represented as a vector $h \in \mathbb{R}^C$. We will also consider PBCs, as this is a natural context for materials simulations[5]. In this context, horizontal translations by $a \in \mathbb{Z}$ units of δx may be described by the function T_a defined as

$$[T_a(x)]_i = x_{i-a}$$

5 The same analysis may be conducted considering other boundary conditions, but boundary points have to be treated differently.

Elements with negative indices are wrapped as a consequence of PBC when reaching the end of the resulting vector. Essentially, T_a moves all elements in the vector a positions down (or up, if $a < 0$), wrapping data from the bottom to the top. In this framework, the following proposition holds:

Proposition 2. *Consider* $x \in \mathbb{R}^C$ *and the affine transformation* $A(x) = \overline{W}x + \vec{b}$, $\overline{W}$ *a square* $C \times C$ *matrix and* $\vec{b} \in \mathbb{R}^C$. *Then* A *commutes with* T_a *if and only if* $\vec{b}$ *is constant and* $\overline{W}$ *is a circulant matrix, i.e. if and only if it is a matrix whose elements on a diagonal are identical and each row may be obtained from the previous one by rotation one position to the right. As an example, a* 4×4 *circulant matrix is in the form:*

$$\begin{bmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{bmatrix}$$

Proof. Proof can be performed by direct calculation applying definitions (notice that proofs for T_1 and T_{-1} suffice, as can be generalized to T_a by induction, as $T_a \circ T_b = T_{a+b}$). □

For other properties of circulant matrices see Ref. [120]. Notice that Proposition 2 implies that a linear equivariant model from $\mathbb{R}^C$ to $\mathbb{R}^C$, *must* have a weight matrix with this special structure. Notice also that circulant matrices offer an important parameter saving, as the number of independent weights in C dimensions is C (as opposed to general matrices, which have C^2 elements). Indeed, this class of matrices can actually be defined in terms of a single row, i.e., they are fully characterized by a single vector in $\mathbb{R}^C$. In Deep Learning slang, the way to say that the same parameters are re-used in many places in the NN structure is that these parameters are *shared*.

Another important result is the following:

Corollary 1. *Consider a NN with* N *layers operating on* $x \in \mathbb{R}^C$, $NN(x) : \mathbb{R}^C \to \mathbb{R}^C$*:*

$$NN(x) = W_N \circ \sigma \circ W_{N-1} \circ \sigma \circ \ldots \circ W_1$$

where σ *is a non-linear, pointwise function, as in Chapter 1,* W_i *are affine transformations and* $\circ$ *denotes function composition. Then* $NN(T_a(x)) = T_a(NN(x))$ *if and only if* ***all*** *linear operators in* W_i *are represented by circulant matrices.*

Proof. Suppose all matrices in W_i in a NN are circulant. Then

$$NN(T_a(\cdot)) = W_N \circ \sigma \circ W_{N-1} \circ \sigma \circ \ldots \circ W_1 \circ T_a$$

By proposition 2, $W_i \circ T_a = T_a \circ W_i$. Commutation $\sigma \circ T_a = T_a \circ \sigma$ is trivial, as σ operates components-wise. The translation operator may therefore be pulled to the front, yielding the result.

On the other hand, suppose that the NN is fully connected and translational equivariant. We prove by induction that all W_i must be represented using circulant matrices. The index N indicates the number of hidden layers, thus $NN_N(T_a(\cdot)) = T_a(NN_N(\cdot))$. The linear model is the base $N = 1$ case,

$NN_1 = W_1$: thesis follows directly by Proposition 2. Suppose (induction step) that

$$W_N \circ \sigma \circ W_{N-1} \circ ... \circ W_1 \circ T_a = T_a \circ W_N \circ \sigma \circ W_{N-1} \circ ... \circ W_1$$

holds, i.e. all matrices W_i, $i \leqslant N$ are circulant. Then this implies that in the case $N+1$:

$$W_{N+1} \circ \sigma \circ W_N \circ ... \circ W_1 \circ T_a = W_{N+1} \circ T_a \circ \sigma W_N \circ ... \circ W_1 = T_a \circ W_{N+1} \circ \sigma W_N \circ ... \circ W_1$$

where the last equality holds by hypothesis. Then this means that W_{N+1} commutes with the translation operator and is therefore circulant by Proposition 2 again. □

This is very nice: if we insert circulant matrices in fully connected NN, we obtain a specialized network with a reduced number of parameters which satisfies translational equivariance of the input vector by construction. Additionally, together with the Theorem 2, we have that these special "circulant networks" have universal approximation capabilities for equivariant mappings.

A key property of circulant matrices is that they may be interpreted as convolution operations. Convolutions are defined for periodic functions f and g as:

$$(g * f)(x) = \int_0^L g(x - x') * f(x') dx' \tag{3.23}$$

being $x \in [0, L]$. If the usual uniform grid of C collocation points is used, this can be approximated as

$$(g * f)_i = \sum_j f_{i-j} g_i \delta x \tag{3.24}$$

where we still consider that negative indices are wrapped. With this, we can draw an important connection: if we extract the (unique up to translations) row vector of a circulant matrix, we may interpret the linear operation it encodes as a discrete convolution operation. Indeed, a more straightforward interpretation of "circulant layers", which now on will be called by their proper name, i.e. *convolutional layers*, is in terms of convolution with a *kernel* [22, 23, 25].

In this picture, elements in the convolution kernel may be identified with interaction terms. One of the main roles of convolution operations in physics is in Green functions [121]. If we look back at Equation 3.23 and consider g as the Green's function of a linear differential operator, we may informally interpret $g(x - x')$ as the effect that a source at x' has at point x (see for example the application to elasticity in Equation 3.17). Similarly, we can consider convolution kernels as (discretizations of) interaction or information mediating functions.

Convolution operations generalize straightforwardly to 2D and 3D cases: the idea is the same, except that convolution kernels and functions are now

represented as 2D and 3D arrays and sliding operation happens in two or three dimensions respectively.

3.2.2 Locality assumption

While convolutions in the form described in Equation 3.24 solve the problem of translational symmetries, we have not discussed how they may be beneficial in lifting limitations related to domain size. Indeed, if we only consider convolutions as an alternative interpretation of circulant matrices, we are still building $\mathbb{R}^C \to \mathbb{R}^C$ mappings. Generalization to different domain sizes, however, may be obtained if a locality assumption is also introduced. Since kernels represent interaction or information mediating functions, this is equivalent to assuming that the value of the convolution output at point x only depends on neighboring points. In terms that are maybe more familiar, this is effectively very similar to cutting off physical interactions (e.g. if x is discretized in space), or a limit in causal relationships in Markov chains (e.g. if x is a time series) [122]. In DL slang, such cutoff radius is a hyperparameter and is referred to as *receptive field*, in analogy to biological structures [23, 25]. Normally, receptive fields are reported in terms of the number of collocation points or pixels for image data.

In practice, only some values in the convolution kernel are different from zero if such a locality assumption is made. This is not only a reduction in the number of parameters, but also allows generalization to larger (or smaller) vectors, as interaction terms for elements further apart than the cutoff radius are by definition zero. Remember that changing the dimensionality of vectors, keeping spatial discretization fixed, is equivalent to changing the domain size: if a Network is *fully Convolutional*, i.e. only contains this kind of local convolutions and pointwise operations, then it can be applied to domains of arbitrary size. Reverting to the circulant matrix formalism, this means that, once the model is trained, the corresponding matrix operating on a larger vector space can be constructed at evaluation time. The price we have to pay is that only local interactions may be represented, reducing the CNN representation capabilities. In practice, however, locality is still a pretty general characteristic for real data applications.

In modern CNN, receptive fields are commonly small odd numbers (3, 5, or 7 are very common [25]). This allows for a further compression of the number of parameters, ultimately leading to a computational advantage. Computational cost reduction, however, is useless if CNNs have too narrow receptive fields. This limitation is partially resolved by the multi-layer structure of NN. This can be understood intuitively with the example sketched in Figure 24. Consider a 2-layer CNN in which convolutions have kernel size 3. This means that values in the first hidden layer at index i will depend on values at $i+1$, i and $i-1$ in the input vector, in the 1D case. As the second convolution also has kernel size 3, the output of the NN at index i will depend on values at $i+1$, i and $i-1$, *of the hidden layer*. This means that the effective NN window is not 3, but 5 instead. This generalizes with subsequent iterations: CNN effective receptive fields scale also with depth.

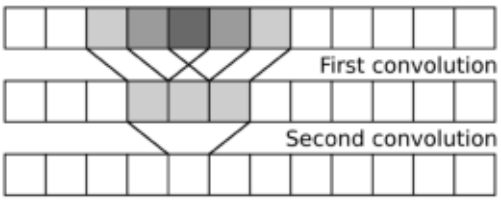

Figure 24: Sketch representing the effective increase of the receptive field with iterated convolutions.

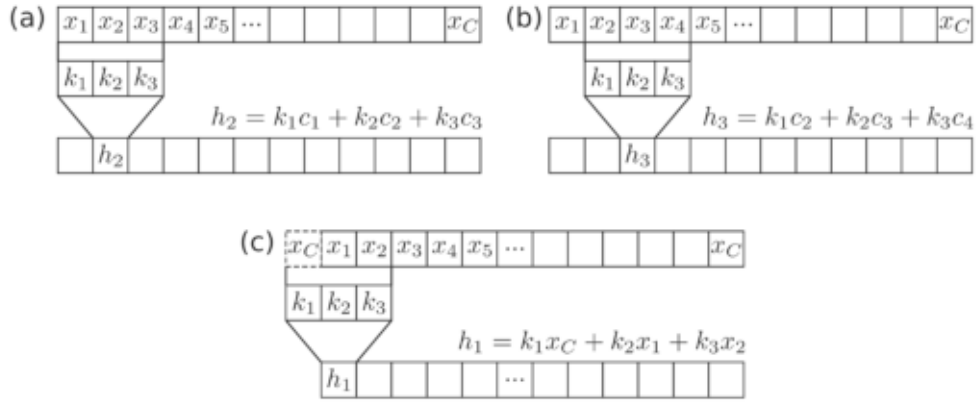

Figure 25: (a,b): example of convolution. The three-element kernel is slid along the input vector x to generate the hidden layer values. In (c) an example of circular padding is shown: when the convolution layer is centered at boundary points, values are "wrapped", effectively implementing PBCs.

For completeness, we mention that there is an additional way to mitigate the locality problem, called *pooling*. The main idea of a pooling layer is that of reducing hidden layers resolution by merging information coming from nearby values (e.g. small patches in images). This is very important in image classification and autoencoder strcutures [22, 23, 25]. In this Thesis, however, these will not be used.

3.2.3 Boundary points and PBCs

Odd-size kernels have the advantage that they may be considered as *centered* at a given collocation point or pixel, providing interactions between nearby elements in the vector. In 1D, the convolution is then calculated in the following way: the kernel is centered at a given point and element-by-element multiplication, followed by addition, is performed. The convolution kernel is then slid one position[6] to the right and the operation is iterated[7], as sketched in Figure 25(a,b) for a 3 element kernel.

From this picture, it is clear that particular care is needed when the kernel is centered at boundary points. To preserve the size of the input vector, augmentation with additional values is required. This operation is referred to as *padding*. Depending on the values inserted, a padding type will be speci-

6 In CNN bigger step sizes, referred to as stride, are sometimes used. This effectively produces a pooling operation (see e.g. [22]).

7 Also, we remark that practical, parallel GPU implementations of this operation are closer to circulant matrix multiplication.

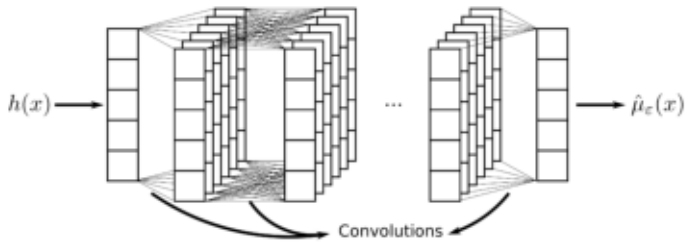

Figure 26: Sketch of the fully Convolutional NN architecture used to predict the elastic chemical potential. Intermediate representations (hidden layers) possess multiple channels. Lines between channels in different layers represent convolutions.

fied. In particular, *constant padding* refers to the condition in which missing elements are assigned a specific value ν (if $\nu = 0$, this is normally called zero padding) and *circular padding* refers to the condition in which missing elements are filled in with copies of the input vector values at the other boundary (see Figure 25(c)). Of course, these are not the only options, and depending on the application other choices may be more suitable. Notice that if no padding is used, the output of the convolution operation is a vector *smaller* than the input one, thus acting as a sort of soft pooling operation. Notice, however, that circular padding is particularly useful in our case, as it encodes periodic boundary conditions by construction (indeed, the PBC and circulant matrix discussion we started with implies circular padding). In applications to physical modeling, therefore, boundary conditions may be inserted naturally in the standard formulation of CNNs: circulant padding encompasses PBCs, special values in constant padding may mark nodes at which Neumann or Dirichlet boundaries are to be applied, etc. [123]

Up to this point, we discussed a single convolution per layer. CNNs in real-world applications are rarely built in this way. In general, many different convolutions operate on the Network input, yielding multiple hidden vectors, as sketched in Figure 26. Following the forward NN pass, the next passage is applying a suitable, non-linear, pointwise operation (this was indicated with σ in Chapter 1). The next hidden layers are customarily also composed of multiple vectors, each of which is obtained as linear combinations of convolutions in the previous layer[8]. The collection of vectors at a given hidden layer is usually referred to as *channels*, in analogy with RGB channels in color images. Indeed, if multiple functions are to be treated at the same time, the CNN input might already be composed of multiple channels. The capability of treating multiple input (and output) channels for a CNN represents another key advantage in computational physics applications: each input/output channel may be assigned to a function in a system of differential equations, for example, allowing the evolution of multiple coupled scalar fields or of vector fields.

8 Translational equivariance is still respected, as translation operations commute with vector addition. Of course, not all hidden vectors are connected to *every* vector in the previous hidden layer. Specialized structures of course proliferate in literature and applications, e.g. Ref. [124].

Name	Symbol	Value
Young Modulus	Y	130 GPa
Poisson ratio	ν	0.27
Ge surface energy	γ_{Ge}	6 eV/nm^2
Si surface energy	γ_{Si}	8.7 eV/nm^2
Wetting d parameter	d	0.27 nm

Table 2: Materials constants used in Morphological evolution simulations.

3.3 PREDICTING THE CHEMICAL POTENTIAL: PROOF OF CONCEPT

Now that the physical model has been described and CNNs have been introduced, it is time to tackle the problem of approximating the elastic contribution to the chemical potential. The profile function $h(x)$ is discretized on a uniform grid of collocation points. Naturally, grid resolution affects the accuracy of predictions based on finite difference schemes. In the specific case at hand, however, a convenient spacing of 1 nm proved to already provide convergent behavior.

Once the the approximate mapping between surface morphology and elastic chemical potential is trained, we can in principle perform faster simulations. The accuracy required from the NN, however, is strict: iterative numerical schemes are required to calculate the free surface evolution from Equation 3.11. For this reason, extrapolation effects must be investigated. In the following, we will therefore start with the simplified task of approximating the Green semi-analytical expression 3.18 first. Once we checked that the NN procedure is set, we will then turn to the full-fledged FEM calculation. This is not the only reason, however. Actually performing millions of FEM calculations may easily translate into days or months of calculations, which does not allow for extensive comparison between CNN predictions and FEM. We will therefore use the Green approximation as a proxy to understand the level of accuracy that can be reached with this method.

We provide in Table 2 the value of materials constant used in simulations. The actual value of D_s is irrelevant as it acts as a multiplier to the timescale, hence we report the used value of $D\ \delta t = 5 \times 10^{-3}$ a.u.

3.3.1 CNN architecture and additional symmetries

As we have discussed in Section 3.2, Convolutional NNs are a natural choice for the task we are considering in this Chapter. Indeed, CNNs already encode equivariance under spatial translations, which is expected from physical considerations in this case.

Some additional adaptations are however required. First, particular care with the locality assumption has to be addressed if we want to perform regression on elasticity-related objects. Indeed, the equilibrium solution of the mechanical problem is in principle a non-local feature (consider for example dislocation stress/strain fields). Very small receptive fields in convolutions may therefore easily result in a CNN that is not capable of identifying longer-range contributions. As such, an architecture with a quite

broad kernel size of 21 colocation points has been chosen. This quantity is comparable with the critical wavelength for ATG in Ge strained films (see Section 3.1.3), but remember that for deep CNN, the effective receptive field is larger than individual convolutions. The overall architecture is built by 5 stacked convolution-tanh blocks, each convolution having 20 channels.

Other symmetries are present in the physical model we are considering in this Chapter: mirror-reflection equivariance and vertical-shift invariance. Since we are assuming an isotropic solid, both from an elastic and a surface energy point of view, the chemical potential should obey the following equality:

$$\mu_\varepsilon(\mathcal{R}(h)) = \mathcal{R}(\mu_\varepsilon(h)) \tag{3.25}$$

where $\mathcal{R}$ represents a mirror reflection operation with respect to the origin[9]. An equivalent result to Corollary 1 should be derived. In the case of mirror symmetry, this leads to symmetric matrices, instead of circulant. In conjunction with the convolutional structure, this yields *symmetric kernels*. This can be enforced by substituting the original kernel $[k_1, k_2, ..., k_n]$ with the symmetrized version $1/2[k_1 + k_n, k_2 + k_{n-1}, ..., k_n + k_1]$, n being the receptive field size, at every forward pass.

Vertical shift invariance is instead related to the fact that the *elastic* chemical potential does not explicitly depend on the film average value. This is not true in general but holds for the Green function small slope limit of Equation 3.18 and in the arbitrary slope case if the substrate and film elastic constants are the same. To encode this symmetry, a simple solution can be adopted and the mean value of $h(x)$ can be subtracted before the NN forward passage.

3.3.2 Training, Validation and Testing

As we already discussed at the beginning of this Section, we will first deal with approximating the cheap Green function approach of Equation 3.18. In order to construct physically sound profiles, a freely available Perlin noise generator [125, 126] has been used. Some additional modifications have been implemented by other group members to generate profiles that also present flat regions. Additional details will be provided in a future publication and for brevity we will not report them here. A dataset composed of $\approx$ 180000 cases has been produced and split into a training and validation set [100]. This is composed of (h, μ_ε) couples on a simulation cell of 100 nm in length (PBCs apply) and with an amplitude ranging from 10^{-3} to 8 nm, where we expect that the small slope approximation is no longer valid.

The training procedure has been performed using the same Adam optimizer [40] discussed for the dislocation case in Section 2.4.3. Similarly, the loss function used is the standard MSE Loss, as this is a regression task. As expected, there is no overfitting, as can be readily observed in Figure 27.

Again, the value of the training loss can be misleading. For this reason, is always a good idea to inspect performances on test cases. Figure 28 shows

9 Actually, the broader class of reflections centered at any point should be considered due to combination with translational equivariance.

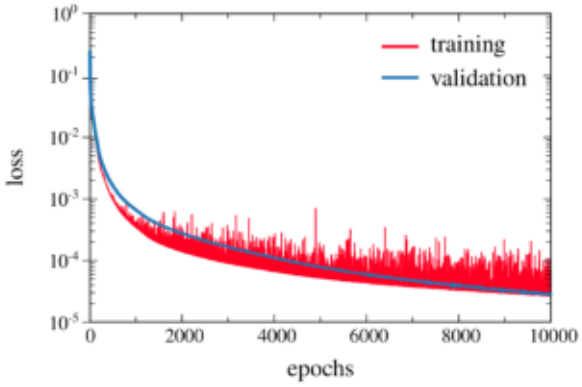

Figure 27: Training (red curve) and validation (blue curve) loss for the CNN applied to the semi-analytical Green function approach. As can be seen from the validation loss, no overfitting is present.

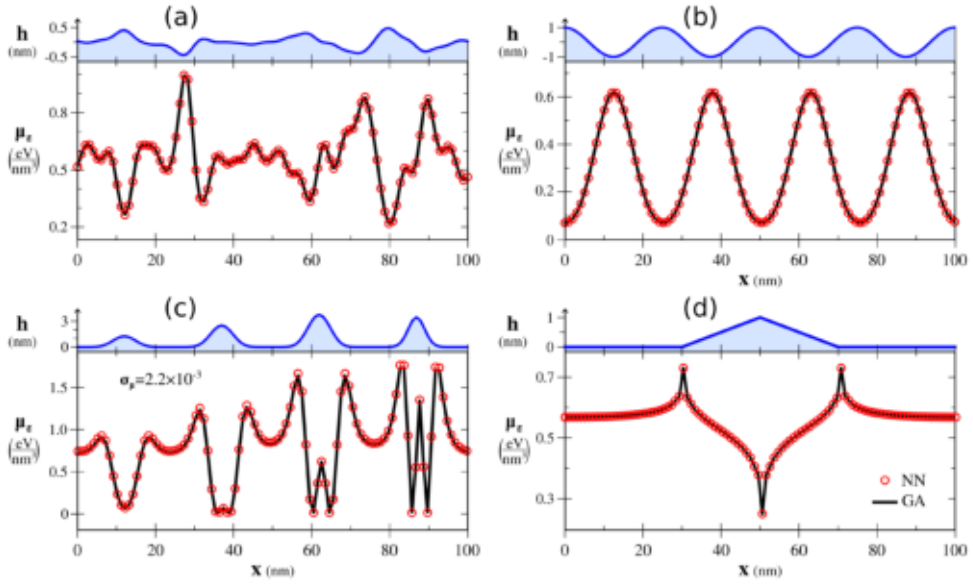

Figure 28: Comparison between NN predicted elastic chemical potential μ_ε and the ground truth value obtained by the Green function approach.

the predicted and true μ_ε for some specific profiles. This suggests that the generalization capabilities of the NN are very good, as one-to-one correspondence is achieved even on profiles that were not present in the training set. It should be stressed that the analytical convenience of some of these profiles is in no way a guarantee that the ML method would provide high-quality approximations. After all, no sharp corners and cusp points were present in the training set. Additionally, CNN approaches are inherently non-linear. This means that their performances *cannot* be predicted by simple Fourier decomposition considerations. Even if the cosine profile of figure 28(b) is present in *all* of the training profiles Fourier transform, it is actually a new profile as any other.

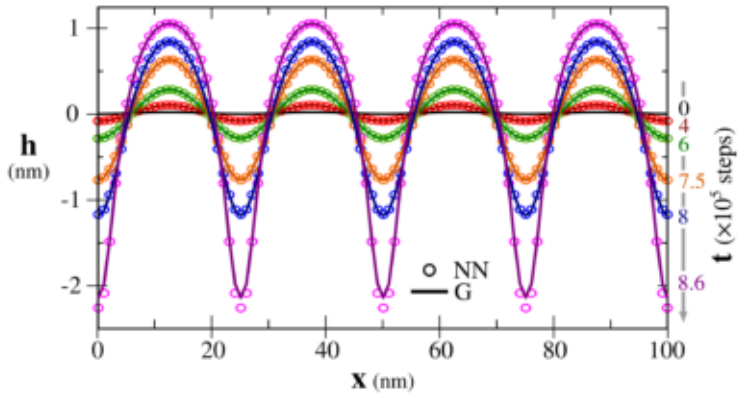

Figure 29: Example of a simulation of ATG behavior starting from a sinusoidal profile. Initial amplitude is set so 10^{-3} nm.

3.3.3 Time integration: ATG from NN

Now that the CNN approximation accuracy has been established, we turn to the question of whether it is capable of providing accurate enough results to also be employed in time integration. As a first test, let us compare the strained film evolution under surface diffusion in the linear stability analysis limit. The main point here is that, if the NN is capable of reproducing the ATG behavior, this is the first confirmation of its predictive capabilities.

A simple forward-Euler integration scheme was used, together with a finite difference numerical implementation for spatial derivatives required for the $\kappa(x)\gamma$ term. The equations of motion are:

$$\frac{\partial h}{\partial t} = \frac{\partial}{\partial x}\frac{1}{\sqrt{1+h'^2}}\frac{\partial \mu}{\partial x} = \frac{\partial}{\partial x}\frac{1}{\sqrt{1+h'^2}}\frac{\partial}{\partial x}[\mu_\gamma(x)+\mu_\varepsilon(x)] \tag{3.26}$$

where μ contains the curvature term and the elastic contribution, but for the moment no wetting term is considered, consistent with the analysis we reported in Section 3.1.3. Figure 29, reports the comparison between the evolution integrated using the NN approximation and the Green (G) approach, with an initial amplitude of 0.1 nm and a wavelength of 25 nm, close to the fastest growing mode of $\approx$ 23 nm [114, 115, 116, 113]. As we are mainly interested in the number of integration steps, we will report time in these terms.

Free-surface intermediate profiles obtained by CNN closely follow the ones calculated from Green's function "true" values, except for the very last stages (see curve in purple in Figure). This is very promising, as it means that the accumulation of errors is small, despite having performed almost 10^6 integration steps.

A single simulation is not enough to consider the CNN method as an alternative for all situations. Instead of looking at individual calculations, we can exploit the fact that in this case there is an analytical expression for the time dependence of the profile function amplitude provided by Equation 3.19 [114, 115, 116, 113]. The same quantity can be extracted from analogous simulations to the one in Figure 29 for different λs. Notice that this implies that the NN should be able to generalize to wavelengths that were

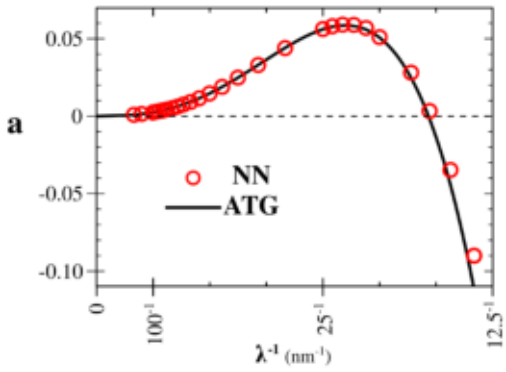

Figure 30: Comparison between the analytical and the amplification factor obtained by time integration using the CNN approximation to calculate the elastic contribution to the chemical potential.

not present in the training set *by construction*. Indeed, as the simulation cells in training examples were 100 nm large, only integer sub-multiples of this value could be present in the Perlin noise profiles. Remember, however, that this is not a problem for a fully convolutional structure: actually, this was one of the main reasons for this choice. This test, therefore, also constitutes a generalization test for at least smaller computational domain sizes. Figure 30 reports the reconstructed ATG amplification factor α curve together with the analytical one from Equation 3.20.

The CNN closely reproduces the true curve. The largest deviations which can be identified are in the very short wavelength regime. However, at short λs the spatial discretization length is more critical and a worse approximation for differential operators should be expected even for traditional methods. We can therefore expect that the NN approach is capable of tackling other tasks successfully.

3.3.4 Time integration: growth simulations

Let us now turn to more complex simulations. An interesting playground is growth and annealing simulations by surface diffusion. The equation describing the system evolution for this case is the following:

$$\begin{aligned}\frac{\partial h}{\partial t} = &\frac{\partial}{\partial x}\frac{1}{\sqrt{1+h'^2}}\frac{\partial \mu}{\partial x} + f = \\ &\frac{\partial}{\partial x}\frac{1}{\sqrt{1+h'^2}}\frac{\partial}{\partial x}[\mu_\gamma(x) + \mu_\varepsilon(x) + \mu_{wet}(x)] + f\end{aligned} \tag{3.27}$$

where μ contains the curvature term, the wetting term, and the elastic contribution to the chemical potential. The new term f represents a vertical deposition flux, yielding the amount of material per unit time and unit area deposited on top of the strained film. In the following, the ratio $f/(D\delta t)$ will be set to 50. To also analyze generalization performances to systems much bigger than the ones considered in the previous Section, we will focus here on simulation cells of 1 µm, 10 times larger than training examples. Figure 31 shows a comparison between simulations performed with the NN approximation and the Green function traditional approach over

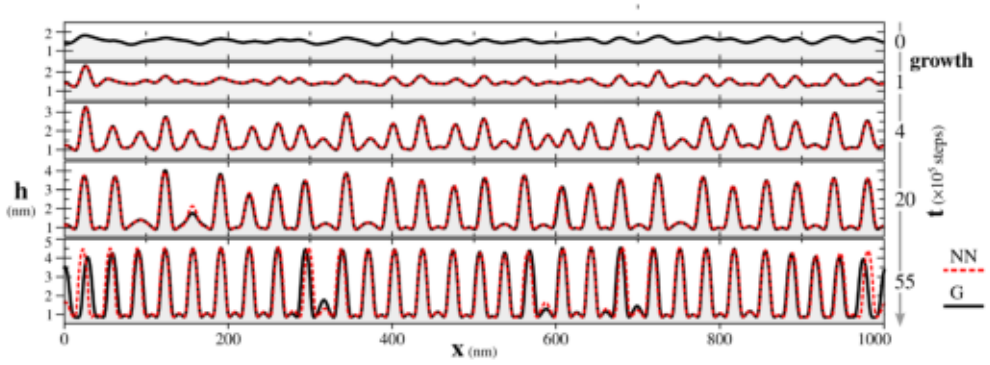

Figure 31: Comparison between growth simulations of a strained film under a constant deposition flux as obtained using the NN approximation (dashed red line) and the Green function semi-analytical approach.

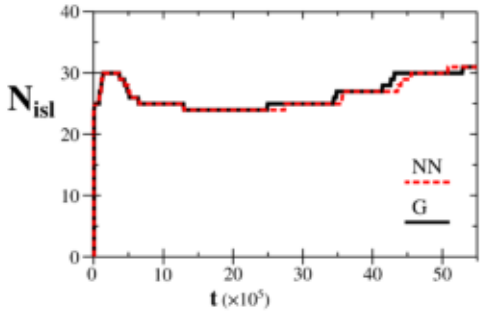

Figure 32: Comparison between the number of islands during the growth evolution of Figure 31. Time is reported in terms of integration steps.

several thousands integration steps. The system morphologies are still very close, albeit one-to-one correspondence is lost for long times. Indeed, this phenomenon should be expected: even if approximation errors are small, after such a large amount of integration steps their accumulation should lead to divergences in the observed profiles.

As the later stages of evolution are affected by this accumulation of small errors, it would be interesting to find an alternative way to compare qualitative features of the surface morphology. Maybe the simplest object that can be considered in this respect is the number of peaks (or Ge islands) in the profile [127]. As $h(x)$ has many local maxima and minima, we only consider peaks that are higher than 1.8 nm to estimate this quantity. A time plot of the number of islands N_{isl} for the evolution in Figure 31 is reported in Figure 32. It can be seen that the number of islands predicted by NN dynamics closely follows the "true" one which was obtained by the Green function approach for the elastic chemical potential. This means that for longer times, while one-to-one correspondence may be lost, at least coarse-grained quantities for $h(x)$ can still be reproduced.

Unfortunately, the results presented in this Section are mostly a proof of concept: the Green function approximation described in the small slope regime is computationally very cheap and there is no need to speed it up. Additionally, it has limitations in reproducing real material behavior. Nonetheless, if we turned directly to expensive FEM calculations, it wouldn't have been possible to easily compare the two time integration procedures.

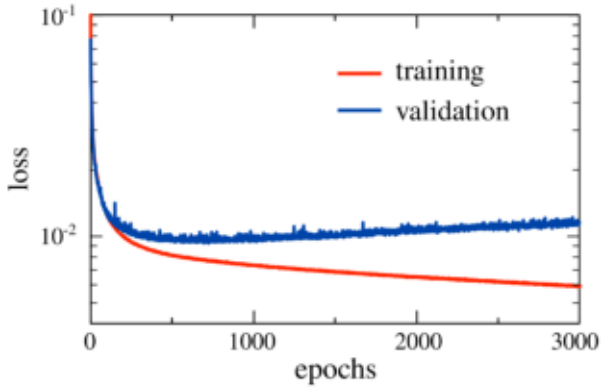

Figure 33: Training and validation loss for a CNN model trained on FEM data. The validation loss increasing after epoch $\approx$ 500 indicates the presence of overfitting.

3.4 BACK TO FEM

Now that we have proven that the CNN approach is capable of providing accurate elastic chemical potentials (at least to the level of qualitative feature reproduction) and that the time integration of these terms is stable, we know that Finite Element calculation efforts will not be wasted. We therefore built a training set through the FEM solver also used in Chapter 2, adapted to the dislocation-free problem. Due to the intrinsically more expensive computational procedure, it would now be unpractical to build a training set as large as the one used in Section 3.3. Approximately 25000 (h, μ_ε) couples were collected, which were then split into a training and a validation set in a 70:30 ratio. Profile functions were obtained through the same Perlin noise sampler, but for a broader amplitude range of $[0.005, 18]$ nm, as there is no reason now to limit to small slope profiles.

To obtain high-accuracy FEM data, a fine discretization of 2000 surface collocation points on the same 100 nm simulation cell is required. Retaining this size for the NN input, however, would be unpractical: in order to maintain the same 21 nm receptive field, convolution kernels would now have $\approx$ 400 parameters each. For this reason, we opted for maintaining the same architecture of Section 3.3 and coarse grain to 100 collocation points FEM calculations. Clearly, a more refined approach would be necessary in this respect and will be a topic for future work.

Figure 33 reports the training and validation losses during training using the same optimizer as for Figure 27. As can be readily observed, after the initial 500 epochs, the validation loss starts to increase while the training one still is decreasing. This is the typical sign of overfitting and is not surprising, since in the current setting we are no longer using such a huge number of examples as in Section 3.3. The model at epoch 500 was therefore used, implementing an early stopping procedure (see discussion at Section 1.8).

As an additional sanity check, we also show chemical potentials for a simple cosine profile (Figure 34(a)) and Perlin noise (Figure 34(b)). In the first case, a one-to-one agreement with the FEM calculation can be observed. This is not the most impressive feature since the example falls in the small slope regime (still remember that it was not present in either training set). What

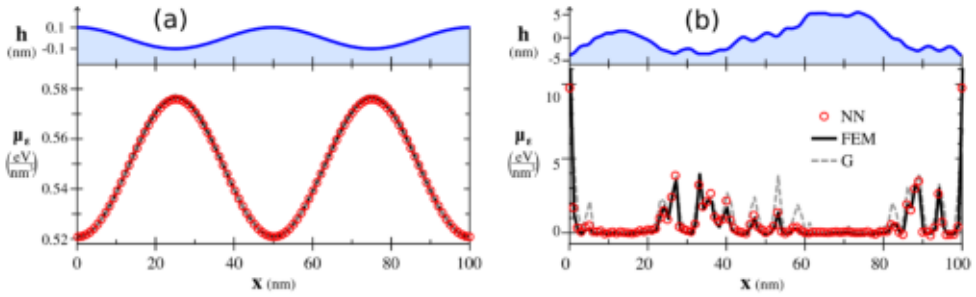

Figure 34: Comparison between NN and FEM predictions on testing cosine and Perlin noise profiles. The value of the Green function corresponding prediction is also reported for reference (not visible on the left due to close agreement).

is more interesting is that the NN approximation also follows the higher amplitude Perlin noise: despite the prediction not being perfect, the CNN approach still provides a much more accurate approximation of the μ_ε obtained through FEM than the small slope Green function. The CNN model is therefore capable of merging the two regimes, treating at one time shallow and very rough morphologies.

Remember that CNN evaluation is very cheap if compared to FEM calculations: in the specific case at hand, without any specialized code optimization beside the standard PyTorch [128] implementation, a computational speed-up of at least 4 orders of magnitude is achieved. We can therefore perform evolution simulations at a fraction of the FEM traditional computational costs retaining a comparable accuracy. As an example, we perform a coarsening simulation with the same parameters used in the previous Section except for the elimination of the deposition flux term. Some snapshots are reported in Figure 35. As we already discussed, it is not possible to directly compare this evolution with a complete FEM simulation. Indeed, reproducing these single dynamics would exceed the time required to build the whole training set by a factor of 200. However, it is possible to compare the NN prediction with one-shot FEM calculations at specific times. These are reported in the left panels of Figure 35. The NN prediction once again closely follows the ground truth. As a final notice, we remark that the end state of the simulation is qualitatively different if CNN or Green function approximation is used. The difference in the size and number of islands is dramatic.

Other simulations are available in the repository [100].

3.5 WHERE SHOULD WE GO FROM HERE AND WHAT IS MISSING

Perspectives for this class of approaches are very promising: with such a huge speed up in calculations and such a high level of accuracy, NN methods could unlock in the future accurate simulations at spatial and time scales which have been traditionally unpractical even for continuum models.

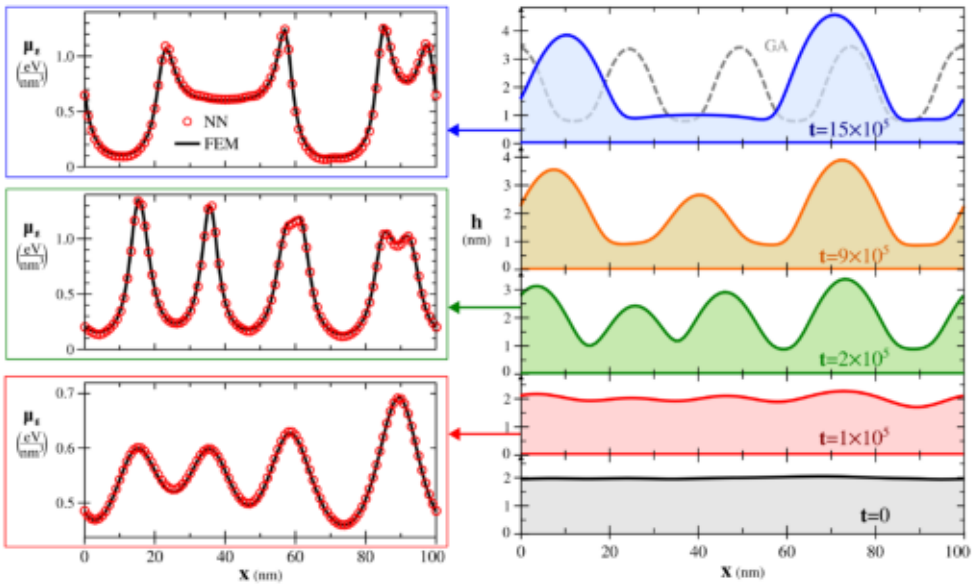

Figure 35: Comparison between FEM and NN predictions for the elastic chemical potential. On the right, the corresponding states during the evolution are reported.

Of course, the application shown here still presents several limitations. First, and most importantly, we are describing only a 1+1D system. A straightforward extension to 2+1D features (see e.g. [129]) could actually be implemented using standard implementations in ML libraries. Unfortunately, computational costs increase sharply with the system dimensionality. The impressive generalization capabilities of the CNN approach used here, however, suggest that maybe even a moderate training set could suffice. Research activity in our group is already moving in this direction.

Another interesting extension could be the generalization to non-isotropic solids, both from an elastic and a surface energy/diffusion point of view.

The last important limitation is related to system resolution. As we have discussed, at least for elastic terms, the receptive fields of NN should be quite broad. This makes the number of parameters required for a CNN very high, which also increases the overfitting risk if the same, simple architecture used here is employed. There are however several more advanced NN architectures that could mitigate this issue, e.g. U-Net [130], pooling layers, encoder-decoder structures [22, 23, 25] or combination of these.

4 CONVOLUTIONAL RECURRENT NN FOR PHASE FIELD METHODS

In this Chapter, we will deal with the possibility of using NN to approximate the time evolution of a system without explicitly needing to integrate numerically a differential equation. This can be done with a specialized architecture, Recurrent Neural Networks, and bypasses one of the main limitations of the approach discussed in Chapter 3, i.e. the requirement to calculate time derivatives of a quantity of interest. This approach will be applied to one of the most flexible materials modeling tools, the Phase Field approach.

The Chapter is organized as follows. Section 4.1 introduces the main concepts of Phase Field models. The traditional derivation of Cahn-Hilliard and Allen-Cahn equations is summarized, together with a discussion on how Phase Field models can be used as an alternative to Mullins-like approaches for surface diffusion. As this is a broad subject we will only outline the main results and leave the interested reader to explore more details in specialized sources, such as Ref. [131].

Section 4.2 outlines how NN specialized to treat time series data can be constructed yielding Recurrent Neural Networks (RNN). Additionally, connections with already discussed architectures and integration with CNN are discussed.

Sections 4.3 and 4.4 apply Recurrent Neural Networks to Phase Field modeling of surface diffusion. Respectively, the first one discusses how Convolutional Recurrent Neural Networks can be adapted to the physical model at hand, while the latter introduces a method to provide a prediction uncertainty estimation.

The last Section collects preliminary results concerning spinodal decomposition, a spontaneous phase separation modeled by the Cahn-Hilliard equation. In particular, Section 4.5 shows how Convolutional Recurrent Neural Networks can be improved to obtain a higher predictive capability and extend results to three-dimensional systems.

Results concerning surface diffusion models are also collected in a publication [132], while spinodal decomposition results are being prepared for future publication.

4.1 PHASE FIELD MODELING

Phase Field (PF) approaches [131] are a class of theoretical and computational methods that constitute a flexible and powerful framework for the description of morphological and microstructural evolution of materials at the continuum level. The main object which specifies the system configuration is a scalar *Phase Field* φ.

Depending on the application, φ may have different physical interpretations. As an intuitive example, consider a solid mixture of chemical species

A and B, in which the coexistence of an A-rich phase and a B-rich phase is possible. Then a reasonable choice for φ is the following:

$$\varphi(\vec{x}) = \frac{1}{2}\frac{c_A(\vec{x}) - c_B(\vec{x})}{c_A(\vec{x}) + c_B(\vec{x})} + \frac{1}{2} \tag{4.1}$$

where c_i is the concentration of chemical species i. With this definition, $\varphi = 1$ corresponds to pure A, $\varphi = 0$ corresponds to pure B and $\varphi = 1/2$ is the value we should expect from an interfacial region in which the relative abundance of A and B are equal. This is not the only possible choice, however. Indeed, an alternative formulation in which φ takes values between 1 and -1 is often used. In the following, however, we will be using the 0/1 coding as it has a straightforward translation in terms of computer vision approaches.

4.1.1 Free energy revisited

Together with the scalar field φ, the other main ingredient of Phase Field models is a *functional* definition of the Free energy $F[\varphi]$. This can be expressed in Ginzburg-Landau functional form [131, 133]:

$$F[\varphi] = \int_\Omega g(\varphi) + k|\vec{\nabla}\varphi|^2 \, d\vec{x} \tag{4.2}$$

where Ω is a suitable domain on which the physical process is defined.

The meaning of $g(\varphi)$ is straightforward: it represents the free energy per unit volume associated with the system if the value of φ is constant throughout the domain Ω. In other words, the function g maps the Phase Field value to the mean-field free energy of the corresponding uniform system. As we will be interested in phase coexistence, a numerical convenient choice is [131]

$$g(\varphi) = a\varphi^2(1-\varphi)^2 \tag{4.3}$$

with $a > 0$. This choice of g presents two degenerate minima for $\varphi = 0$ and $\varphi = 1$, representing the coexistence of phases. A more rigorous treatment through the theory of regular solutions would prove the existence of such a double-well potential, albeit with a different functional form [133].

Naturally, the mean-field term cannot tell the whole story. Indeed, in a closed system with an equal number of A and B atoms, it is impossible to obtain uniform $\varphi = 1$ or $\varphi = 0$ values. It can be shown that in such a system, below some critical temperature spontaneous phase segregation happens [134]. Remember, however, that we are searching for a field variable that should interpolate between the two phases continuously. For this reason, an additional free energy term $k|\vec{\nabla}\varphi|^2$, proportional to the gradient of the Phase Field is introduced. $k > 0$ is for the moment just some constant weighting of this "gradient cost". Notice that if this term is removed, it would be possible to have a φ function which oscillates wildly from point to point despite having a zero total free energy, *whatever* is the mean com-

position of the system. As this term arises in transition regions from A-rich and B-rich phases, it controls the *interfacial energy*.

This intuitive introduction to Phase Field free energy functionals already highlights some of the advantages of this class of methods. First, they are very flexible, as couplings with additional fields and physical models can be done by adding other contributions to the free energy functional. For example, if one phase is strained with respect to the other, then Equation 4.2 can be simply augmented with an additional elastic energy term (e.g. see Ref. [135]). Second, PF models are very general, as the chemical species concentration case we discussed so far is only an example. In principle, any *order parameter* can be chosen as φ: magnetization, density, local lattice parameters, etc. can be converted into suitable Phase Fields. In section 4.1.3 we will discuss how φ can be used for an implicit representation of a material domain.

Another favorable feature is the possibility of dealing with complex geometries. Indeed, a Phase Field may also be used to represent regions in a computational domain and there is a specialized formalism in which such auxiliary fields may be used to impose boundary conditions to other differential equations [103]. On a high-level perspective, one or more additional Phase Fields may be used to constrain the free energy functional to subsets of the computational domain Ω by "tagging" such regions. This is particularly well suited in cases in which the geometry of boundary conditions is very complex and changes as a function of time or as an effect of the differential equation solutions themselves (so-called *moving boundary problems*). We will exploit this fact to perform surface diffusion-driven morphological evolution simulations.

4.1.2 Chemical potential in Phase Field models

Now that the free energy functional has been introduced, we can derive the driving force determining the time evolution of φ, allowing for the modeling of non-equilibrium processes. The idea is fundamentally analogous to the one we have seen in Section 3.1: the (generalized) chemical potential can be obtained as the functional derivative of F with respect to the Phase Field

$$\mu = \frac{\delta F}{\delta \varphi} = g'(\varphi) - 2k\nabla^2\varphi \tag{4.4}$$

which can be directly obtained by Euler-Lagrange equations [110].

Equation 4.4 allows us to have more insight into the physical meaning of the k constant in the gradient energy term. If we consider the minimization of free energy, this amounts to solving the non-linear, second-order partial differential equation $\mu = 0$. This can be solved analytically in one dimension if expression 4.3 is used for g. Imposing boundary conditions for $\lim_{x\to\pm\infty} \varphi(x)$ which represent pure A on the left and pure B phases on the right, we obtain:

$$\varphi(x) = \frac{1}{2}\left[1 - \tanh\left(\sqrt{\frac{a}{4k}}(x - x_0)\right)\right] \tag{4.5}$$

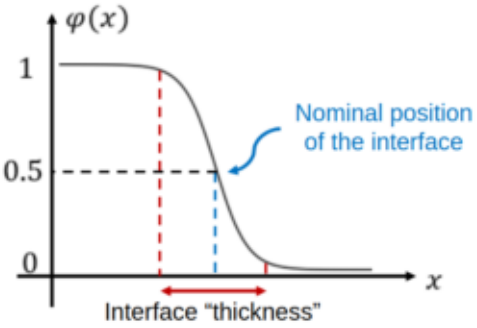

Figure 36: Equilibrium configuration of φ in 1D. The phase field interpolates continuously between $\varphi = 1$ and $\varphi = 0$ phases. The nominal position of the interface is at $\varphi = \frac{1}{2}$

where x_0 defines the nominal position of the interface between the A- and B-rich phases. Notice that this identifies the ratio $\sqrt{a/k}$ as the controlling parameter of the interface "smearing": if $a \gg k$, then the hyperbolic tangent solution tends to a Heaviside function and we recover a *sharp interface* picture. Unfortunately, such limit is not always easy to consider in simulations, hence Phase Field models are regarded as examples of *diffuse interface models*, in contrast with *sharp interface* approaches such as the Mullins' one we discussed in Chapter 3.

Equation 4.5 also provides a more physical interpretation of the parameters in the free energy functional. By definition, the interaction energy between an A-rich phase and a B-rich phase can be simply expressed removing the bulk contributions to the total free energy

$$F[\varphi] - (g(1)V_A + g(0)V_B) = \gamma S \tag{4.6}$$

where V_A and V_B represent the volume of the corresponding phases and S is the interfacial area. In one dimension, we can plug into the equation the solution 4.5, yielding a connection between PF model parameters and the interfacial energy γ:

$$\gamma = \frac{\sqrt{ka}}{3} \tag{4.7}$$

Notice that this equivalence with γ only fixes one of the two parameters k and a.

4.1.3 From free energy to evolution equations

Now that the chemical potential is defined, it is possible to derive equations of motion for φ. In this Thesis, we will mainly deal with two models for Phase Field approaches: evolution of conserved order parameters (so-called *model B*) and PF approaches describing morphological evolution by surface diffusion. For completeness, however, we will also report equations for non-conserved Phase Field (so-called *model A*) as derivation is straightforward.

As we have defined in the previous Section, the driving force for the evolution is given by $\mu = \delta F/\delta\varphi$. If there is no conservation of the order param-

eter (e.g. it represents local magnetization, order/disorder transitions, etc. see [131]), we can consider simple dissipative dynamics: the rate at which φ changes is inversely proportional to the local chemical potential

$$\frac{\partial \varphi}{\partial t} = -M\mu \tag{4.8}$$

where we introduced a kinetic constant M. Inserting only the terms considered in the previous Section in the free energy, we obtain the so-called *Allen-Cahn equation*. Other physical contributions with respect to interface energy may be inserted in $F[\varphi]$ if their dependence on the Phase Field is known.

The case of conserved order parameters (e.g. composition fields) has to be treated with a bit more care. By definition, we should impose φ local conservation. In differential form:

$$\frac{\partial \varphi}{\partial t} = -\vec{\nabla} \cdot \vec{J}_\varphi \tag{4.9}$$

where $\vec{J}_\varphi$ is the flux associated with φ transport. This latter quantity may be related to gradients in the chemical potential utilizing Onsager linear laws [112]:

$$\vec{J}_\varphi = -M\vec{\nabla}\mu \tag{4.10}$$

where the coefficient M represents now a diffusion constant. Assembling this last relation with the conservation of mass, we obtain

$$\frac{\partial \varphi}{\partial t} = \vec{\nabla} \cdot M\vec{\nabla}\mu = M\nabla^2\mu \tag{4.11}$$

Notice that the last equality is possible only if M does not depend on φ or the position explicitly, e.g. in the case of isotropic and homogeneous diffusion processes. If the functional form discussed in Section 4.1.1 is used, Equation 4.11 becomes the well-known *Cahn-Hilliard equation*. One of its most famous applications is to the study of the so-called *spinodal decomposition* phenomenon, which occurs in solid mixtures [134] (usually metals) and polymeric materials [136]. Remember that the similarity of Equation 4.11 with the simple Fick diffusion law is only formal: μ itself contains second-order derivatives and is non-linear in φ. Numerical solutions costs for Cahn-Hilliard and Allen-Cahn models strongly depend on additional terms in the definition of the system free energy: while standard formulations have been heavily studied, couplings with more complex terms may be much more expensive.

4.1.4 Phase Field models for surface diffusion

As a last case, we outline in this section how a specific modification of the Cahn-Hilliard model can be turned into a PF representation for surface diffusion. If φ is used to implicitly track a solid geometry, instead of some concentration field, then $\varphi = 1$ correspond to bulk points, $\varphi = 0$ to points out-

side the material and the free surface should be identified with the $\varphi = 1/2$ isoline.

At this point, we should ask why this approach should be taken if the traditional Mullins approach is already successful in modeling morphological evolution and growth of solids without the need to smear out the free surface, as discussed in Chapter 3. When the solid free surface may be described by a simple profile function $h(x, z)$, sharp interface models are indeed very convenient. This is not, however, always the case. For example, under some conditions, a material domain may split or merge with a second object. At this point, if a parametric curve/surface describes the surface, specialized procedures have to be considered. In phase field models, on the other hand, φ is defined everywhere in the computational domain, making the handling of such topological changes automatic.

Staying on a very high-level perspective, if φ marks the presence of atoms in a computational domain, we should *confine* their motion on the surface of the solid itself and conserve their total number, if we want to recover a surface diffusion model. This is reminiscent of a "surface" version of the Cahn-Hilliard equation. It is reasonable, at least in the case of an isotropic solid, to consider the following expression for the evolution of the Phase Field:

$$\frac{\partial \varphi}{\partial t} = \vec{\nabla} \cdot M(\varphi) \vec{\nabla} \mu \tag{4.12}$$

where $M(\varphi)$ is a "localizing" function that has a maximum at $\varphi = 1/2$ and vanishes for bulk points ($\varphi = 1$) and far outside the solid where $\varphi = 0$. Due to its form, the equation is sometimes called the *degenerate Cahn-Hilliard* model. A popular choice for the function $M(\varphi)$ and for the free energy functional is the following [103]:

$$\begin{aligned} M(\varphi) &= M_s \frac{36}{\eta} \varphi^2 (1-\varphi)^2 \\ g(\varphi) &= \frac{18}{\eta} \varphi^2 (1-\varphi)^2 \\ k &= \frac{\eta}{2} \end{aligned} \tag{4.13}$$

where η is an interface thickness parameter and M_s is a surface mobility constant. Notice that the isotropic surface energy γ has been here implicitly set to unity.

The main idea is that, as $\eta \to 0$, the evolution equation for the $\varphi = 1/2$ isoline converges to the corresponding sharp interface model. In this case, for very small interface parameters Equation 3.11 should be recovered [103].

4.2 RECURRENT NEURAL NETWORKS

It is necessary at this point to explain why we need machine learning at all in dealing with Phase Field models for Materials Science. Indeed, time integration of the Cahn-Hilliard or Allen-Cahn equation, per se, is not such a computationally expensive task. In this respect, however, if more complex

physics is inserted in the model, limitations similar to the solution of the elastic equilibrium problem in Chapter 2 and Chapter 3 arise. On the other hand, surface diffusion models can be computationally challenging on their own: to have a close reproduction of surface-localized dynamics, small interface thickness parameters η should be employed, which in turn asks for fine, adaptive FEM meshes for accurate tracking of the evolution. This is particularly critical when the more complex case of anisotropic surface energies is considered, as dependence on the specific crystalline facet also increases the degree of the PDE. All in all, the extensive study of realistic systems is once again hindered by the computational speed of some approaches.

In this Chapter, we want to explore a new direction in which NN methods can provide some help with this problem. While in Chapter 3 we used DL techniques to provide a cheap approximation of driving force terms in the equations of motion, here we want to attack the issue from a different angle. Indeed, there are NN approaches that analyze and generate sequential data directly. The idea is, therefore, to use these algorithms to bypass the requirement of small time advancements to reconstruct the system dynamics: as Deep Learning methods are inherently non-linear and universal approximators, they can in principle provide the mapping between the state of the system at some time t and the state at $t + \tau$, with τ being potentially a huge timestep if compared with traditional time integration schemes. Another important consideration is that time relations in sequence elements are not required to satisfy a differential equation at all for these NN methods to be effective. In principle, the results we will discuss here should generalize to experimental images directly. Of course, in that case, the main bottleneck would be the collection of a large enough dataset.

We will now introduce *Recurrent Neural Networks* (RNN) to deal with time sequences. We will do that by considering a simple problem in which a sequence of scalar values x_t, with integer $t \in [0, 1, ..., T]$ denoting a discrete time index, is used as a dataset. T defines the total length of the system. In principle, if we have no information on the kind of relationship between x values, we might be tempted to use a feed-forward NN to approximate the relationship:

$$x_1 = \mathrm{NN}[x_0|\vartheta] \tag{4.14}$$

This is in some sense what we have done in the previous Chapter: we exploited a (Convolutional) NN to approximate the mapping between the state of the system at time t and the state after an infinitesimal interval δt. Notice, however, that we also used a lot of implicit assumptions on the nature of the relationship. For instance, we know that $x_{t+\delta t}$ only depends on x_t and not on the state of the system at previous stages (as we were considering a first-order differential equation). What if, instead, we are dealing with data that don't share this property? We need some specialized NN architecture that is capable of handling a variable number of inputs:

$$x_t = \mathrm{NN}[x_{t-1}, x_{t-2}, ..., x_0|\vartheta] \tag{4.15}$$

A similar problem arises if we consider a classification task for sequences. The NN should be able to have a sequence of arbitrary length as an input,

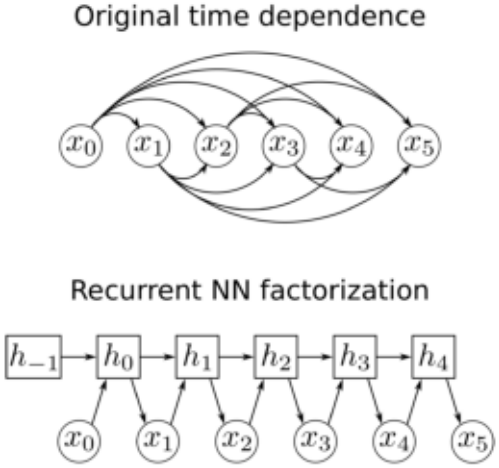

Figure 37: Skecth of a simple RNN for sequence prediction. In the original time sequence, each value of x_t depends on the state of the system at *all* previous times (top). In the RNN reformulation, the system is augmented by the introduction of the additional hidden state variable, which mediates dependencies from previous states (bottom).

pretty much in the same way as fully convolutional structures can generalize to images or arrays that are larger than the one used during training.

Indeed, convolutional structures sound like a better idea than simple fully connected networks: after all, a sequence of scalars is an ordered array, which was the premise of CNNs. Unfortunately, however, these approaches may struggle because of two problems. The first one is related to the translational equivariance assumption, which may not be verified by all sequences. The second is that long-time relationships require the use of very broad receptive fields (in principle even infinite), which might be unpractical in many situations.

4.2.1 Introduction to Recurrent NN

The main idea behind Recurrent Neural Networks [22, 25] is to factorize time dependencies in a different way introducing an additional variable h_t, called a *hidden state*[1]. Notice that despite the name, this has nothing to do with hidden layers in NN. The idea is that h_t in some sense contains information on previous states of the system and x_t can therefore be obtained as

$$x_t = f(h_t|\vartheta_f) \tag{4.16}$$

Where f is a function to be learned (as the parametric dependence on ϑ_f suggests, we will use a Neural Network as a model). Now, it may seem that we are playing with semantics here and just performed a substitution $x_{t-1}, x_{t-2}, \ldots = h_t$, but this is not the case. First, h_t is a vector, or another multi-dimensional equivalent, with a *fixed* size. This means that a simple

1 The reader should not be worried for possible confusions with the profile function $h(x)$ used in the previous Chapter, as it will not be used anymore now on.

fully-connected structure could in principle be used for f without problems. Second, the hidden state h_t only depends on x_t and h_{t-1}:

$$h_t = g(h_{t-1}, x_{t-1}|\vartheta_g) \tag{4.17}$$

for a suitable function g. It is at this point needless to say that g too will be parametrized with a Neural Network.

The main advantage of this reformulation is that x_t can be obtained by repeating applications of f and g functions *irrespective of the sequence length*. This iterative use of the same function is the reason why these structures are called *Recurrent* NN. At the same time, passing through a hidden state allows the propagation of information from possibly very remote sequence elements up to the present. From a heuristic point of view, h_t may be considered as a *memory* of previous states of the system. f no longer needs to access states at previous times, as long as g has learned how to extract and retain from them important information.

Tackling the problem from a probabilistic perspective allows for a more systematic treatment. Let us turn back to probabilistic models (see Section 1.3.1. Training a NN model that generates sequences involves the maximization (through MLE or MAP) of the probability of observing the training set element:

$$\mathbb{P}[x_0, x_1, ..., x_{T-1}, x_T] = \mathbb{P}[\{x_t; t \in [0, T]\}] \tag{4.18}$$

in Chapter 2 and 3, we could assume that training set elements were extracted independently from the same distribution. This still holds for whole sequences, but *elements* in the same time sequence are *not independent*. Using basic probability theory [20], we can write

$$\mathbb{P}[\{x_t; t \in [0, T]\}] = \mathbb{P}[x_{t'}|\{x_t; t \in [0, T], t \neq t'\}]\mathbb{P}[x_{t'}] \tag{4.19}$$

for some chosen time index t. As future events do not influence the past (at least in physical models we will consider), we also have:

$$\mathbb{P}[x_{t'}|\{x_t; t \in [0, T], t \neq t'\}] = \mathbb{P}[x_{t'}|\{x_t; t < t'\}] \tag{4.20}$$

At this point, we may start from x_0 and iteratively unpack the probability of observing a given sequence in the following form [22]:

$$\mathbb{P}[x_0, x_1, ..., x_T] = \prod_{t'=0}^{T} \mathbb{P}[x_{t'}|\{x_t; t < t'\}] \tag{4.21}$$

notice that this holds for generic sequences. This is the formal version of the "different length input" problem that we introduced at the beginning of the Section: in principle, we should have a different conditional probabilistic model for each timestep! This is a big issue, especially if we want to generalize to timesteps that are outside the training set range.

If the system is "memory-less", a possible solution to the problem is to use the Markov property. Formally, it states that the conditional probability of observing the state of the system at time t', only depends on some subset

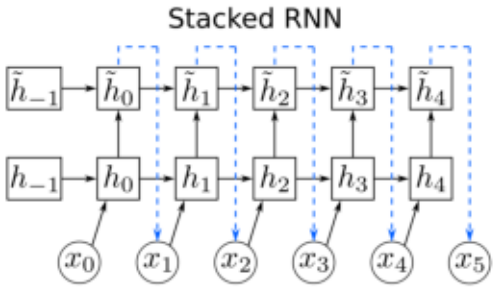

Figure 38: Example of the increase of RNN capacity through stacking of two recurrent blocks. Hidden states are termed h and $\tilde{h}$ respectively. $\tilde{h} \to x$ connections are reported with dashed blue lines for visual clarity.

of previous states. For first-order Markov processes (i.e. processes that only depend on the previous state of the system):

$$\mathbb{P}[x_{t'}|\{x_t; t < t'\}] = \mathbb{P}[x_{t'}|x_{t'-1}] \tag{4.22}$$

As long as the transition probability does not depend explicitly on time, this permits the use of the same probabilistic model for all transitions. Indeed, this would end up in a similar training procedure to the one which we considered in the previous Chapter, with the quirk that input and output are the same quantity measured at different times.

Unfortunately, Markov property is not universal. In principle, then, one should consider the full description of Equation 4.21. The introduction of the hidden state, however, bypasses this problem. In practice, we perform the substitution:

$$\mathbb{P}[x_{t'}|\{x_t; t < t'\}] = \mathbb{P}[x_{t'}|h_{t'-1}] \tag{4.23}$$

In this context, the hidden state h may be regarded as an additional random variable that allows for the decoupling of the state of the system at time t' and the past (see sketch in Figure 37). The crucial point is that h possesses the Markov property:

$$\mathbb{P}[h_{t'}|\{h_t; t < t'\}, \{x_t; t < t'\}] = \mathbb{P}[h_{t'}|h_{t'-1}, x_{t'-1}] \tag{4.24}$$

Notice that in principle nothing is preventing $h_{t'}$ from having a degenerate probability distribution, e.g. a deterministic mapping, as we are not interested in the probability of h itself. After all, the hidden state is an unobservable quantity. In practice, the use of h allows one to recast the probability of a (possibly) non-Markov process in a form that resembles a Markov one. In other words, we can interpret the hidden state as an augmentation of the base state x with a specific conditional probability structure. Notice also that nothing is preventing such a hidden state model to capture fully Markovian behaviors.

Before moving to practical training of RNN, let us discuss a point regarding how the complexity of these models can be increased. In principle, the main way of augmenting an RNN model capacity should be the increase in the number of elements in the hidden state, i.e. its *width*. Similarly to other

NN approaches, however, most of the time increases in depth proved to be more efficient. It is thus possible to *increase the number of hidden states* by stacking multiple RNNs on top of each other and using the hidden state of one module as the input of the following one (see Figure 38).

4.2.2 Training RNNs

In order to train an RNN, we must build a loss function which is once again defined in terms of MLE/MAP. In particular, if we are performing regression on sequence values, the loss function reads:

$$\mathcal{L}(\vartheta) = \sum_{t=t_{start}}^{T} \mathcal{L}_t(\vartheta) \tag{4.25}$$

where ϑ is the set containing both g parameters ϑ_g and f parameters ϑ_f. Notice that the summation is performed only for values between the initial time at which the RNN starts to generate a sequence t_{start} and the final time available in training T. Notice that the initial time *need not* to be zero, as it is in general possible that a Recurrent network takes a sub-sequence before starting to generate subsequent elements. Still, the RNN needs *some input* to begin with: at least the initial value for the hidden state (h_{-1} in our notation, usually set to a zero vector) and the value of x_0 should be provided externally to the model to start the recurrence.

For sequence regression we can take $\mathcal{L}_t(\vartheta)$ as:

$$\begin{aligned} &\mathcal{L}_t(\vartheta) = (\hat{x}_t - x_t)^2 = (f(h_t|\vartheta_f) - x_t)^2 \\ &h_t = g(h_{t-1}, \hat{x}_{t-1}|\vartheta_g) \end{aligned} \tag{4.26}$$

where $\hat{x}$ is the NN prediction, x represents the training set value. Notice that the *generated* previous step has been used to calculate h_t, in consistency with evaluation time in which ground truth values are no longer available. To arrive to the squared loss term we made the usual (although not unique) choice of a probabilistic model in which x_t is conditionally normally distributed:

$$\mathbb{P}[x_t|h_t] = \mathcal{N}(\hat{x}_t, \sigma) = \mathcal{N}(f(h_t|\vartheta_f), \sigma) \tag{4.27}$$

with unspecified standard deviation σ.

Optimization requires gradient calculations at different times. Notice, however, that $\mathcal{L}_t$ requires $\hat{x}(t)$, which in turn can be calculated only if h and $\hat{x}$ at previous times are also computed. During the backward pass, it is therefore possible to re-use gradients of $\mathcal{L}_t$ for previous times. For this reason, the backpropagation algorithm for RNN is termed *backpropagation through time*.

Another important point in the practical training of RNN for sequence regression is related to the length of the input sequence. Sticking to the approximation of Phase Field models, it is clear that the number of input states must be at least equal to the order of the underlying differential equation. In our case, dynamics is first order, hence a single initial snapshot should

be sufficient to predict the system evolution. We remark that this fact is *not equivalent*, in principle, to a pure Markov property: a RNN which requires a single initial state x_0 can still use *all* intermediate stages (through the hidden states) to generate the state at time t. It is also clear from the Loss function 4.25 that RNN performs a regression task which targets *the whole sequence at once* instead of the individual state-to-state transition as would happen under a full Markovian assumption.

Providing a RNN with a single (or the minimal) amount of snapshots, however, may result in very slow training, especially if it is required to generate very long sequences. At the beginning of optimization, weights will be mostly random and the loss function and the iterative generation of a sequence will be severely affected by accumulation of errors. One possibility to circumvent this inconvenience is to provide the NN with the "correct" time sequence (from the training set) for all timesteps, so that error accumulation is reduced. Loss function terms $\mathcal{L}_t(\vartheta)$ are still calculated as in Equation 4.26, but in this case the hidden state calculation is modified as

$$h_t = g[h_{t-1}, x_{t-1}|\vartheta_g] \tag{4.28}$$

where the *true* previous state x_{t-1} has been used instead of the predicted one $\hat{x}_{t-1}$. This procedure is called *teacher forcing*, in analogy with a student (the network) being corrected at every question by a teacher (training data) to avoid the accumulation of previous mistakes during an exam. This seemingly innocent substitution has a profound impact: we drastically changed the information flow from past states to future ones, as h_t is no longer calculated on previously generated elements! This is critical, as once the model is used to predict new sequences, ground truth examples will no longer be available: small fluctuations in predicted outputs can quickly accumulate and make the NN prediction unreliable. Indeed, there is a fundamental difference between Network inputs that come from the user (e.g. training/validation set inputs) and those that have been generated by the RNN itself and fed back into the Network. RNNs in which the latter type of input-output connection is present are said to have *output recurrence* [22].

For once, it is possible to both reduce computational costs and end up with a model trained on the same input-output connectivity which will be used at evaluation time. This procedure is called *curriculum learning* and is based on the idea that at the beginning of training, where the computational speed up is more important, it is better to employ *teacher forcing*. In later stages, however, full output recurrence is used, so that the final model is fine-tuned to learn the correct information flow. In intermediate epochs, the first part of the sequence may be passed in teacher forcing fashion and the remaining part be generated in closed-loop mode. In some sense, during the transition between the two regimes, the training task gets progressively more complex, hence the name *curriculum*. The optimal transition schedule is once again problem-dependent and can be considered as an additional training hyperparameter.

Regression is not the only task that can be tackled using RNNs. The other typical Machine Learning application is that of classification. In this respect, Recurrent Networks represent a flexible framework to perform the classifi-

cation of entire sequences of arbitrary length [22, 23, 25]. Here, we will only outline the simple case in which the whole sequence is associated with a single class, although more complex scenarios are possible. Once again, a probabilistic model is required, hence the critical object is the conditional probability:

$$\mathbb{P}[\text{class}|\{x_t; t \in [0, T]\}] = \mathbb{P}[\text{class}|h_T] \tag{4.29}$$

where we simply recovered the same hidden state assumption and condition on h_T instead. In practice, sequence classification proceeds producing a hidden state associated with the whole sequence h_T. This is then fed into a second network which converts it in class probabilities. This can be done in teacher forcing without any problem, as the Network is never used in closed loop.

4.2.3 Gated Recurrent Units

From the discussion in the previous Section, it seems that there is no further complication to RNN other than using enough parameters for a flexible enough network for g and f, while carefully handling conditional probabilities and input-output relationships. However, when long-time dependencies are present, this is easier said than done. Indeed, the recurrent application of the same network calls for *vanishing and exploding gradient* problems [22, 23, 25].

This phenomenon can be understood through a simple example. Consider a situation in which some quantity y (e.g. a sequence element or a hidden state) is obtained by recurrent application of some linear model, parametrized with a linear transformation $\overline{W}$. Then

$$y_t = \overline{W}^t y_0 \tag{4.30}$$

It is a basic result in linear algebra that eigenvalues of $\overline{W}^t$ are powers of eigenvalues of $\overline{W}$. Unless all of them have magnitude exactly one, a pretty rare situation, an exponential dependence (either growing or shrinking) on parameters in $\overline{W}$ is present. This makes the training process problematic, with possibly infinitesimally small or diverging optimization steps.

In the case of RNNs, the same argument may be made, albeit the presence of non-linear applications has to be considered. Indeed, during backpropagation in time, the same matrix recurrent application problem arises, both in terms of affine transforms and in terms of jacobians for non-linear applications [137, 22].

To solve this problem, specialized architectures have been proposed over the years. One of the most successful strategies is the so-called Long Short Term Memory (LSTM) Network developed by Hochreiter and Schmidhuber [137]. A more modern version, which also has a slicker architecture is the Gated Recurrent Unit (GRU) [138]. The core idea in both cases is the careful use of non-linearities to implement a gating mechanism in past to future mappings. This solves the vanishing and exploding gradient problem and stabilizes training.

We will now briefly discuss the mathematical form of the GRU, as it will serve as the core implementation of RNN in the following applications[2]. To calculate the next hidden state of the system h_t, first the so-called *reset* gate value is calculated:

$$r_t = \sigma(W_r x_{t-1} + U_r h_{t-1}) \tag{4.31}$$

where W_r and U_r represent affine transformations, while σ is *fixed* to be the logistic sigmoid of Equation 1.25. r_t is a vector whose components are in $(0, 1)$ and with the same dimensionality of the hidden state. Once r_t has been calculated, the tentative new hidden state $\tilde{h}_t$ is defined as

$$\tilde{h}_t = \tanh\left(W x_{t-1} + U(r_t \odot h_{t-1})\right) \tag{4.32}$$

where W and U represent other affine transformations. The symbol $\odot$ indicates instead Hadamard product (element-wise multiplication). This equation clarifies at least on a high level the role of r_t: acting as a multiplicative gate, it controls how much information should be used from the previous hidden state to generate the next one: a reset gate value of 0 corresponds to removing data "stored" in the memory, while a value of 1 represents full memory retention. Before obtaining the effective hidden state, however, a similar but independent *update gate* is defined

$$z_t = \sigma(W_z x_{t-1} + U_z h_{t-1}) \tag{4.33}$$

The actual hidden state is then calculated by

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{4.34}$$

In other words, the final hidden state is a weighted average of the previous value h_{t-1} and the proposed new value $\tilde{h}_t$. In this respect, the update gate controls how much information has to be re-written inside the RNN "memory", closing the control of information flow from previous steps. Notice that the GRU (as the LSTM) operates on array components: it is completely possible that the reset gate has a value of 1 in a specific element and is 0 on another, allowing for higher flexibility for the model and to have multiple time-scales in the same architecture.

4.2.4 Convolutional Recurrent Neural Networks

As we stated at the beginning of this Chapter, our main goal is to apply RNN to Phase Field modeling. It is clear, however, that the spatial structure of this kind of data calls for some properties that we already discovered are easily encoded by Convolutional Neural Networks. In other words, for 2D PF simulations, we would like to implement a Recurrent NN which operates on *sequences of images*. How can we choose between the two architectures?

2 Notice that time indexing may be inconsistent with other sources, as we are already specializing the algorithm for series generation, in which the distinction between the RNN output at the previous time (usually indicated with y_{t-1}) and the present input (usually denoted with x_t) is here immaterial. We therefore mix the two notations using x_{t-1} to denote the state of the system at the previous step

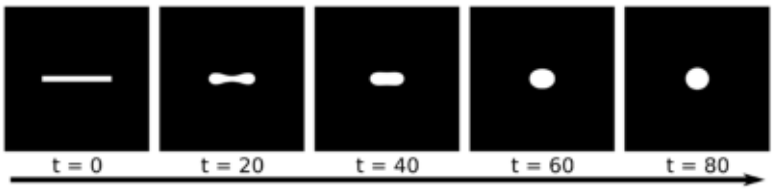

Figure 39: Example of an evolution by surface diffusion of a simple rod-like domain as obtained through FEM solver. The final state is circular, consistent with the sharp interface model. t is measured in terms of time separation between training set snapshots.

The simple answer is that we don't! Indeed, we exploit one of the most powerful features of DL: modularity. The idea was proposed first by Cho et al. [139] for LSTM and then extended to GRU by Ballas et al. [140]. When the prediction of image sequences is required, we simply substitute general affine transformations in GRU equations with convolutions. After all, we have proven in Section 3.2 that CNNs can be interpreted as fully connected NN which comply by construction to translational equivariance and locality inductive biases. It is therefore, at least from an abstract point of view, straightforward to generalize GRU to (fully) Convolutional RNN (CRNN) by simply promoting hidden states from vectors to images or volumetric data. The hidden state width is as a consequence substituted with the number of channels in these representations. In this way, it is possible to obtain a NN which is capable of processing and generating sequences of arbitrary length which are composed by images of arbitrary size. Indeed, the composition of these qualities is being shown to be very appealing for materials science applications and is currently an active field of research [13, 132, 141].

4.3 SURFACE DIFFUSION BY MACHINE LEARNING

We will now turn to train a CRNN to approximate morphological evolution by surface diffusion in 2D. This means that φ implicitly defines the boundary line between the "interior" and the "exterior" of a 2D material domain. Under the driving force of free surface minimization, the typical evolution of an elongated domain passes through intermediate "hourglass" shapes and ends in a circular profile, as also predicted by the sharp interface corresponding model (see Figure 39).

4.3.1 Training set construction

Exploiting the group's historical experience, we therefore built a training and validation set composed of 1200 and 250 sequences respectively. Data have been obtained through high-resolution FEM simulation using the Adaptive MultiDimensional Simulation toolbox (AMDiS) [142, 143]. This software is particularly well suited for this class of calculations, as it employs an automatic, efficient and adaptive meshing procedure to track regions in which φ changes more rapidly, i.e. free surfaces, thus containing computational

Figure 40: Samples for initial conditions in the training set.

costs. The PF model implemented is the one defined in Section 4.1.4, using an interface thickness parameter $\eta = 1/400$ the simulation box length and a local mesh refinement so that the FEM element is $\approx 1/7\eta$ near free surfaces.

High-accuracy simulation of evolution by surface diffusion may be computationally expensive, especially if the considered domains have long perimeters (2D) or large surface areas (3D). For this reason, the dataset was composed of simple, prototypical shapes, such as the ones reported in Figure 40. Different configurations were obtained varying the length of rectangles, keeping the short side equal to 8 in units of η. Once obtained, FEM snapshots have been converted to .png format with a resolution 90×90. Notice that pixels are much larger than FEM elements, which is one of the main reasons, together with the much longer timestep (250 times larger), which allows for computational speed up.

Results and additional details can also be found in our publication at Ref. [132].

4.3.2 Additional symmetries

As in the sharp interface case, convolutional layers are not enough to capture all symmetries present in the system. For the problem at hand, there are at least three other classes of operations to be considered:

- The degenerate Cahn-Hilliard model is by construction conservative.
- Since we are modeling isotropic materials, the system is also equivariant with respect to reflections and rotations for arbitrary angles.
- Due to the specific choice of the free energy functional and mobility $M(\varphi$, evolution is equivariant with respect to the transformation $\varphi \to (1 - \varphi)$. Physically, this means that pore evolution is isomorphic to that of material domains.

We now describe some strategies we employed to try and solve these issues.

Regarding order parameter conservation, for the moment we will use a naive solution, while in Section 4.5 we will discuss a more advanced method. An additional penalty term is added to the loss function, increasing the cost for sequences in which the spatial integral of φ deviates from the one of the original sequence:

$$\mathcal{L}_{\text{cons},t}(\vartheta) = \left(\langle \varphi(\vec{x}_t) \rangle - \langle \tilde{\varphi}_t(\vec{x}|\vartheta) \rangle\right)^2 \tag{4.35}$$

where $\langle \cdot \rangle$ indicates spatial integration, φ_t is the true field obtained by FEM simulations and $\tilde{\varphi}_t$ is the RNN output at time t. Explicit dependence on the

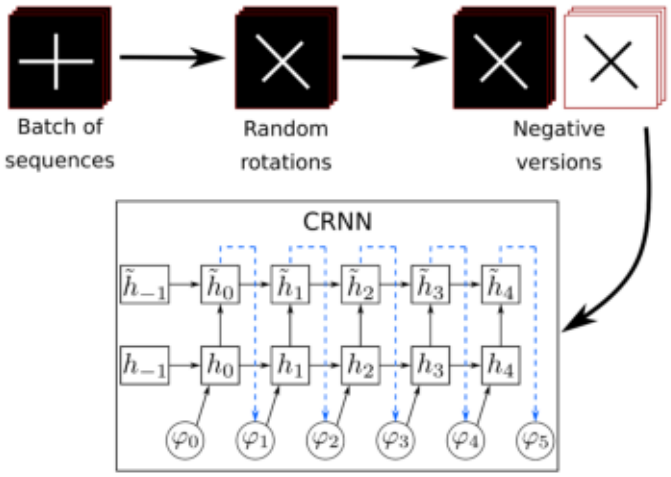

Figure 41: Sketch representing the training flow. Each evolution in a batch is first randomly rotated, and then the transformed version $\varphi \to (1-\varphi)$ is generated. Data are then submitted to the CRNN (bottom). Each arrow in the CRNN structure corresponds to multiple convolutions.

previous states of the system or from the hidden state has been suppressed for clarity.

With the additional conservation term, the total loss function is

$$\mathcal{L}(\vartheta) = \frac{1}{T}\sum_{t=t_0}^{T} \mathcal{L}_t(\vartheta) + \lambda \mathcal{L}_{\text{cons},t}(\vartheta) \tag{4.36}$$

where λ is the hyperparameter controlling the relative weight of the loss function terms. The term $\mathcal{L}_t$ corresponds to the simple MLE term defined in Equation 4.26. If we interpret the total loss function in terms of MAP, notice that the conservation term $\mathcal{L}_{\text{cons},t}$ arises from the relaxation of the simplifying assumption that the prior is independent from the input variable (see Section 1.6).

Regarding reflection and rotation symmetries, in principle, one could use the same argument of Corollary 1 to find out specific properties for the convolutional kernels. While this is straightforward for reflection operations, for continuous rotation this is not the same, as they constitute an infinite symmetry group. We therefore opted for *data augmentation*: every sequence is rotated by a random angle before being processed by the CRNN. This ensures a sampling of the overall configuration space without the computational costs of actually performing FEM simulations and contains the time required for each epoch during training. Notice that this operation is possible only because the shapes in the training examples are isolated shapes in a uniform background.

The last symmetry to be considered is the inversion $\varphi \to (1-\varphi)$. Again, one in principle could use the corresponding version of Corollary 1 and find that constraints are now on the activation function and the bias terms in affine transformations. Unfortunately, as we are using a Gated Recurrent Unit, we have no freedom of choice in for activation functions. We therefore resort again to data augmentation. This time, as the symmetry group being

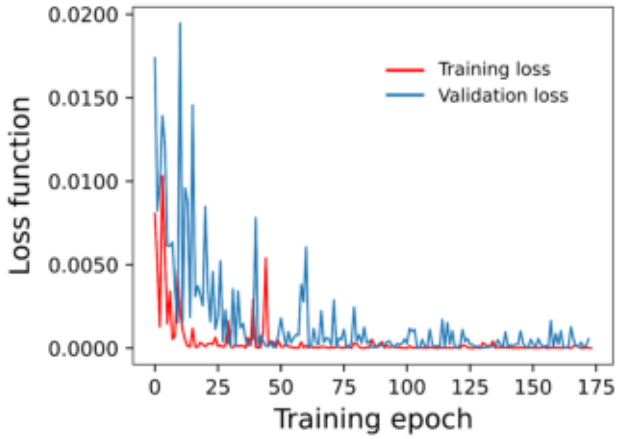

Figure 42: Training and validation loss for the CRNN used to approximate the Phase Field evolutions by surface evolution.

considered comprises a single operation, we will submit to the CRNN every example in both the "positive" and "negative" versions. The procedure is equivalent to averaging the gradients as calculated on the material domain evolution and the pore one, effectively canceling biased contributions.

A sketch of the overall training procedure is shown in Figure 41.

4.3.3 Training procedure

Armed with the concepts discussed so far, we have everything that is needed for training. For completeness, we report the hyperparameters used. Additional details can be found in Ref. [132].

The training was performed for 175 epochs. The transition between teacher forcing and full output recurrence happened in 25 epochs through linear interpolation of the number of provided ground truth examples (between 149 and 1). The resulting loss plot is shown in Figure 42. Despite the high number of parameters ($\approx 2.8 \times 10^5$), there is no overfitting evidence, although the loss function oscillates more than what was observed in other training procedures shown in previous Chapters. This is to be expected, as the prediction of a full evolution given an initial subsequence is a much more complex and sensitive task than simple state-to-state mappings.

As a confirmation of the quality of the generated trajectories, Figure 43 shows the comparison for a test set case. One-to-one correspondence between the predicted and ground truth Phase Field states may be observed. Importantly the RNN was provided only the initial state of the system, suggesting that this kind of result may be achievable without any further FEM calculations, as a partial teacher forcing in evaluation would have been required. The computational speed up is about three orders of magnitudes (seconds vs hours on the same workstation).

4.3.4 Evolution results

Now that we assessed that RNNs are capable of providing accurate evolutions for validation cases, we will test generalization capabilities. As a first

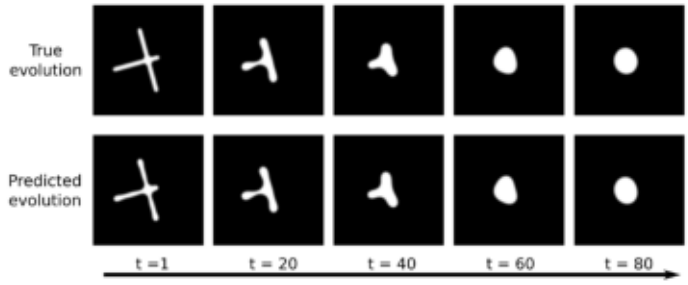

Figure 43: Evolution comparison between CRNN solver and FEM as discretized on the 90×90 grid for a validation case.

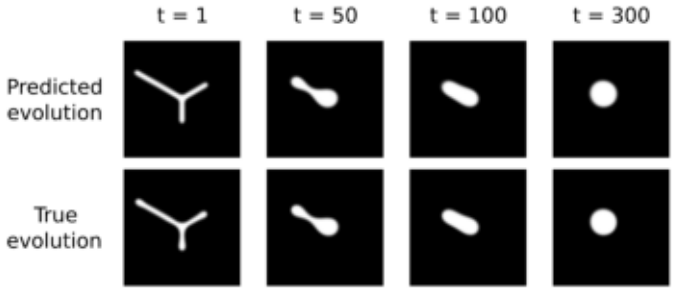

Figure 44: Evolution comparison between CRNN prediction and FEM solver as discretized on the 90×90 grid for a test case. The model was not exposed to this kind of initial condition and was trained on shorter sequences (150 snapshots).

assessment, we show here testing on a qualitatively different initial condition with respect to the ones shown to the CRNN in training. Again, only the initial condition was provided to the CRNN. Some snapshots are reported in Figure 44. The one-to-one correspondence between the ground truth evolution and the generated one is still present. Notice also that this evolution is not only a generalization test in terms of previously unobserved initial conditions but also testifies that the Network is capable of providing time sequences longer than the ones observed in training (here 300 timesteps vs the 150 used in training).

As a further test of these qualities, we also report the evolution for initial elliptical conditions. This is a critical test, as all the training and testing cases discussed so far were composed of rod-like domains with a fixed thickness. Results are in Figure 45. The test is again a generalization check in time, as sequences 3 times longer than training set elements were generated. The CRNN approach exhibits close correspondence with the ground truth evolution, showing that scale behaviors have been learned to some degree. This example also demonstrates another strong point of CRNN approaches: while the elliptical domain evolution in training was simulated independently, the ML prediction was performed in a single domain containing all shapes. This is possible due to the fully convolutional nature of the architecture utilized and is equivalent to the domain size generalization shown also in Chapter 3.

To prove that the accordance is not only qualitative, we also report the time dependence of the domains aspect ratios as a function of time in Figure 46 for the smallest (panel a) and the largest (panel b) elliptical domain

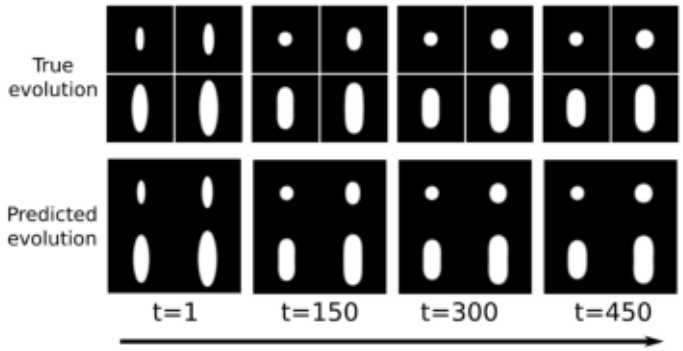

Figure 45: Evolution comparison between CRNN prediction and FEM evolution discretized on the 90×90 grid for elliptical domains. Notice that all the domains are in the same computational cell for the ML prediction.

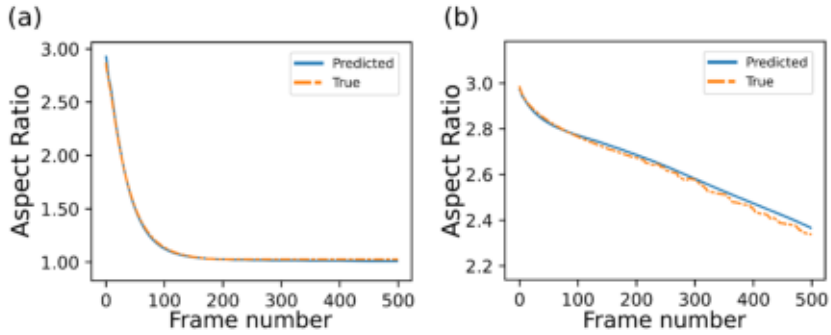

Figure 46: Comparison between the time evolution of the aspect ratio of initially elliptical domains as predicted by the CRNN and by FEM. (a) refers to the smallest ellipse in Figure 45, (b) to the largest one.

respectively. This proves that predictions are not only qualitative but also quantitative.

4.4 PREDICTION UNCERTAINTY ESTIMATION

One of the most important drawbacks of Neural Networks is that when they are applied to new inputs it is not possible to assess the uncertainty on the prediction they provide, at least in the framework we have been using up to this point. This problem is actually an implication of Maximum Likelihood (or MAP): during training, we are only considering one set of parameters which (locally) maximize the probability of observing the training set. While validation and testing procedures are a way to assess generalization error, they do it only in a very broad sense: after validation, we know the *expectation* for deviations from the predicted value. An additional problem is that validation and testing are *finite sets*. It is, therefore, possible that at "running time", i.e. in a real-world environment, the Network is faced with data that not only were not present in the training set but are also outside the region in which validation/testing was performed.

It is therefore crucial to obtain a prediction uncertainty procedure that can assess at run time how unreliable a NN output is. Of course, the only

way to know the prediction error on a sample is to actually calculate the ground truth and make a comparison. Clearly, this cannot be the solution: in the end, we are using ML models to avoid making expensive traditional calculations. Fortunately, there exist statistical methods that overcome, at least to some extent, this limitation of the Maximum Likelihood criterion.

Among various possibilities, the use of ensemble methods [22] is one of the most straightforward. The core concept is that, instead of training a single NN, multiple copies of the same architectures are trained in parallel. Once the ensemble has been trained, individual models should provide similar losses on training and validation/testing and predictions should be close together. However, as training was performed independently, there is no guarantee that this will happen. If predictions for new input data are spread out, this is an indication that the uncertainty is high. There is however an important caveat in this kind of procedure: the spread of the prediction can only be an *estimate* of the prediction uncertainty.

4.4.1 Bootstrap, Bagging and Prediction uncertainty

There are different methods to obtain an ensemble of predictors. Some of these leverage specialized NN layers, such as Dropout [144] to train a stochastic network which effectively provides a different set of weights every time the forward calculation is done. Here, however, we will implement a more traditional approach based on a *bootstrap procedure*. Importantly, this not only allows one to have an estimation of the prediction uncertainty [145] but also makes for a more robust combined predictor [146].

As stated, the core concept in ensemble methods is to train independent models with the same architecture. An easy way to ensure that the training procedure will converge to a different minimum in the Loss function is to build a corresponding number of independent training sets. From a statistical point of view, if we want to train M models, this amounts to the extraction of N samples from the data probability distribution (in our case the distribution of snapshots obtained by FEM integration of the Phase Field model of surface diffusion) a total of M times. Unfortunately, if data availability is scarce, this is unpractical and we have only one training set available.

Bootstrap is a procedure based on the construction of multiple training sets starting from the original one. From a practical point of view, this is done by simply replacing the data distributions (the one containing *all possible PF evolutions* for us) with the original training set itself. Extraction *with replacement* of N elements M times allows to have at least an approximation of the desired independent training sets. The new training sets generated in this way are called *bootstrap samples*.

Bagging exploits the bootstrap procedure to build an aggregate, more robust predictor. The idea is that bootstrap samples can be used to train multiple models. After training, the prediction on new data is taken as the average between all the individual NN predictions. It is possible to prove that the expected deviation from this average is smaller than the average deviation of individual models. Formally, if the (x, y) is the input-output couple and

D_i indicates the bootstrap sample, it can be proven from straightforward statistical arguments that

$$\mathbb{E}_D(y - \psi_i(x))^2 \geqslant (y - \psi_A(x))^2 \tag{4.37}$$

where ψ_i is the predictor trained on the i-th bootstrap sample and $\psi_A(x)$ is the aggregation $\frac{1}{M}\sum_i \psi_i(x)$. The difference between the two values depends on how much the training procedure is influenced by changes in the bootstrap set used. This method is therefore expected to be particularly well suited for Neural Networks, as proliferation of local minima in the Loss function and a dependence from random weight initialization is expected.

What about the prediction uncertainty? This quantity can be simply estimated through the calculation of the standard deviation of the individual model predictions, as we are dealing with a regression task. In practice, it is possible to simply consider the quantity [145]

$$\sigma_D = \sqrt{\mathbb{E}_i(\psi_i(x) - \psi_A(x))^2} \tag{4.38}$$

Notice that for the Phase Field models we are considering in this Thesis, the predicted quantity is a sequence of images. The standard deviation between individual predictions can therefore be calculated as a *function of position and time*: implementing this simple bootstrap procedure we can calculate *when* and *where* the aggregate prediction is expected to be unreliable. In the next section, we will show some applications for this method.

4.4.2 Aggregating models for Surface Diffusion

We now show some application of the bagging approach. Specifically, a total of 15 bootstrap sets have been generated, so that the final aggregate model also averages and calculates standard deviation over 15 different sequences.

A test for the prediction uncertainty estimation has been performed on what, from a physical point of view should be a stationary state for the system, i.e. donut-shape configurations. This can be understood considering the corresponding sharp interface model: as perimeter curvature is constant everywhere, the gradients in chemical potential should be zero and the profile should not evolve. The CRNN aggregate model shows however something different and the central hole is filled with material, although it takes several timesteps with respect to the length of sequences provided in training to observe this phenomenon. The states predicted by the model ensemble are reported in Figure 47. Remember that the models were not exposed to non-evolving states in training, however. Despite the aggregate evolution not being close to the true, stationary one after $\approx$ 500 timesteps, the ensemble standard deviation readily identifies unreliable prediction regions, i.e. errors due to extrapolation.

As a final test, which also exploits spatial domain generalization capabilities, we report a large domain simulation in Figure 48. Clearly, the predictive capabilities of the CRNN are very limited, as the local prediction uncertainty indicator increases in the first evolution steps. This is expected, as the training set did not involve any interacting domains. Still, these evolutions are

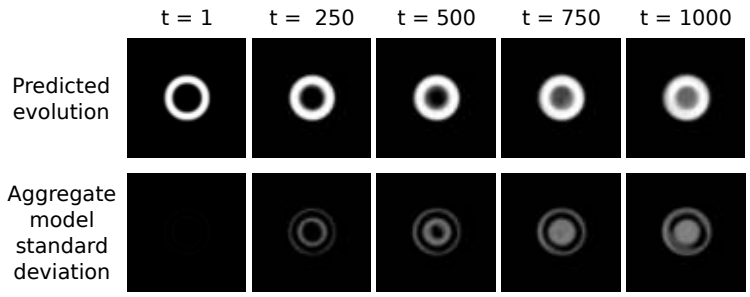

Figure 47: CRNN ensemble prediction for the "evolution" of a donut-like stationary state (top). The aggregate model fails to correctly predict that the system configuration is stable. The prediction uncertainty estimation (bottom), however, readily identifies the extrapolation regime.

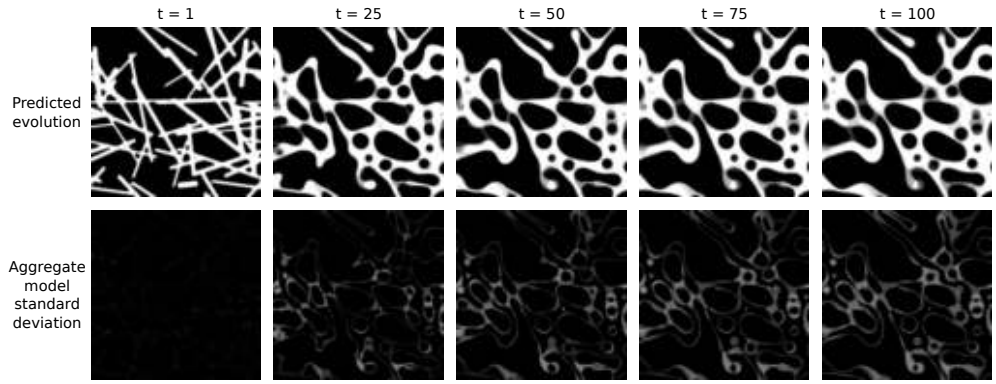

Figure 48: Large size system evolution containing multiple interacting materials domain as predicted by the CRNN ensemble and its prediction uncertainty estimation.

interesting: on one hand, where σ_D is small the predicted evolution is in agreement with the general trends that are expected in surface evolution dynamics, i.e. the rounding of the free surfaces. Secondly, the uncertainty estimation procedure readily identifies that regions in which the CRNN has provided unreliable dynamics correspond to areas in which drastic topological changes (domain merging and splitting) take place. Again, this is expected, as the training set did not contain information on the PF model behavior in such regimes.

A more subtle additional source of error accumulation can be identified: in the initial condition at $t = 1$ it can be observed that the spatial separation between domains is in some cases smaller than individual pixels. Since the CRNN is elaborating a coarse-grained version of the original PF model, this type of initial condition has to be dealt with carefully. Indeed, some modifications to the convolutional structure of the network (again, UNet-like architectures [130], which allows for non-linear resolution compression) may become critical in future development of this approach. An alternative could be exploiting the Recurrent aspect of the CRNN: if longer excerpts of sequences are passed as initial conditions, this could help the NN to resolve ambiguities arising from spatial coarsening and direct it to a more accurate prediction. These directions will be certainly considered in future works.

4.5 TOWARDS 3D MODELS

Some improvements to the CRNN framework have been developed recently in the group. While results are being collected and are not yet submitted for publication, we retain that they are very promising and therefore we discuss some of these in the present Section. To our knowledge, this is the first application of such CRNN schemes in 3D to Materials Science problems.

As 3D simulations are more challenging from a computational point of view, we will deal with the simpler spinodal decomposition model described by Cahn-Hilliard Equation 4.11, the main advantage being that accurate tracking of the interfacial regions between the two phases is less critical. This is because φ variations are no longer localized, as the scalar mobility M implies bulk diffusion. Simulations have been performed using a simple finite difference scheme and an explicit forward-Euler time integration. A total of 550 sequences composed of 50 snapshots each have been produced, with initial conditions initialized by the usual Perlin noise [125, 126], which has been straightforwardly adapted to 3D. Domain resolution is $64 \times 64 \times 64$ collocation points. Periodic boundary conditions have been employed, as they can be readily implemented in CRNN via circular padding. Due to the isotropic nature of the mobility constant, reflections and 90° rotations about all axes also define symmetry operations available for data augmentation. For completeness sake, we report parameters: $k = 1.5$, $a = 6$ and the integration step defined by $M\delta t = 1.25 \times 10^{-3}$. A snapshot was saved every 800 Euler integration steps, which will be the reported unit of time.

4.5.1 CRNN model improvements

The main modifications to the architecture as described in previous Sections can be divided into two different contributions. The first one regards the non-linear nature of recurrent structures in the gating mechanism in the convolutional GRU. As stated in Section 1.7, it is often more effective to increase the depth of a NN, i.e. the number of subsequent stacked layers, than increasing the width of the hidden layer representation. For convolutional NN, width corresponds to the number of convolution kernels to be used and ultimately it is related to the number of channels in hidden states. Recent tests on the spinodal decomposition dataset revealed that inserting additional convolutional layers in GRU modules calculating the reset r, update z and suggested next hidden state $\tilde{h}$ (see Section 4.2.3) allows for a strong compression in the number of parameters of approximately a factor of 10. This is mainly ascribed to a reduction in the number of channels. Importantly, this *increases* prediction capabilities.

The second improvement is related to the $h_t \rightarrow \hat{\varphi}_t$ mapping. One of the main downsides in the implementation discussed for surface diffusion is that the conservation of order parameter was enforced only in a *soft* way via the additional loss term. This, however, does not ensure exact conservation. A simple addition of a hand-crafted convolutional layer can however enforce this condition by construction. As we will discuss, the specific values to be inserted in the convolution kernels can be derived based on physical con-

siderations, highlighting the important interplay between NN architectures and the underlying physical models.

The conservation equation for a scalar quantity can be written in differential form as in Equation 4.9 which we report here for convenience:

$$\frac{\partial \varphi}{\partial t} = -\vec{\nabla} \cdot \vec{J}_{\varphi}$$

i.e. the time derivative of the order parameter is equal to the divergence of a vector field $\vec{J}_{\varphi}$ representing a density flux. As we are representing Phase Field values on a discrete grid, divergence $\vec{\nabla}\cdot$ may be represented as a finite difference operator. Critically, this is equivalent to the summation of the result of convolution operations. For example, in 2D, the derivative with respect to x can be written in two dimensions as a convolution with the kernel:

$$\Delta_x = \frac{1}{\delta x} \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.39}$$

where δx is the collocation point spacing in the x direction. Generalization to three (or higher dimensions) is straightforward but more cumbersome in notation. The main point is that the CRNN may be employed to predict a N-channel map instead of the Phase Field value directly, where N is the spatial dimensionality of the system. Every channel represents one component of the predicted vector field $\hat{\vec{J}}_t$. The variation in the Phase Field value $\Delta_t \varphi_t$ may therefore be computed via application of the finite difference convolutions as

$$\Delta_t \hat{\varphi}_t = \Delta_x \hat{J}_{x,t} + \Delta_y \hat{J}_{y,t} + \Delta_z \hat{J}_{z,t} \tag{4.40}$$

where we directly provided the 3D form. Notice that, if circular padding (or PBCs to use the physics community terminology) are employed, the linearity in the convolution operator implies that the spatial sum of values in $\Delta_t \varphi$ vanishes *exactly*[3].

The predicted subsequent order parameter field is therefore obtained in a forward-Euler fashion as

$$\hat{\varphi}_t = \hat{\varphi}_{t-1} + \Delta_t \hat{\varphi}_t \tag{4.41}$$

which conserves the spatial integral of $\hat{\varphi}$ at all t by construction. Notice that, despite the resemblance with a first-order explicit integration scheme, this is more powerful, as the variation $\Delta_t \hat{\varphi}(t)$ has been obtained through the CRNN under non-Markovian assumptions. As a side note, we remark that this structure may be regarded as a specialized Recurrent ResNet block [147] (see also Section 5.2.2).

3 This is actually true for all convolutions whose kernels element sum is zero, as can be readily proven by direct calculation. More general operators with desired conservation properties can in principle be constructed using this fact.

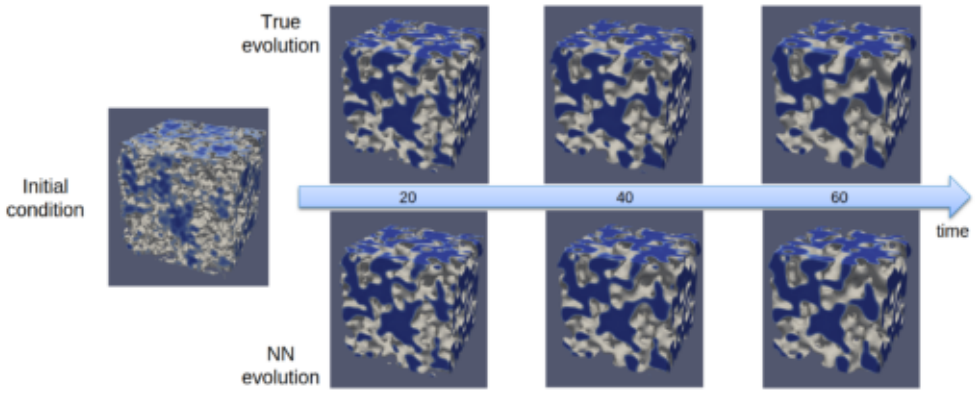

Figure 49: Comparison between spinodal decomposition from time integration of Cahn-Hilliard equation by finite differences with the one provided by CRNN for a test case. Time is reported in terms of NN sequence elements. Only $\varphi \leqslant 1/2$ is shown.

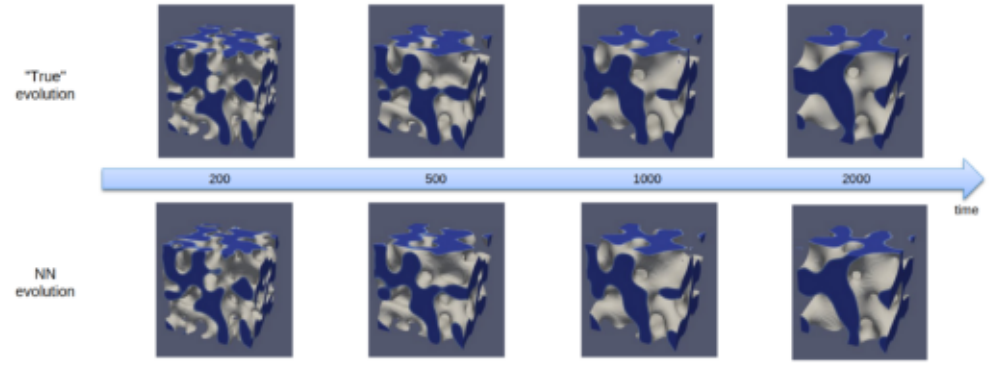

Figure 50: Comparison between spinodal decomposition patterns from time integration of Cahn-Hilliard equation and those provided by CRNN for a number of snapshots 40 times larger than what was used in training. One-to-one correspondence is retained.

4.5.2 Spinodal decomposition by CRNN

We now show some results regarding applications of this improved scheme to 3D spinodal decomposition. First of all, we show a testing case in Figure 49. There is a close agreement between the predicted sequence and the ground truth simulation. This test also involves spatial generalization, as the total volume is 8 times the one provided in training examples. To better appreciate patterns, only $\varphi \leqslant 1/2$ is shown.

This test suggests that the generalization capabilities of the 3D CRNN are high. To prove this, we show in Figure 50 an example of time generalization that exceeds by 40 times the length of sequence provided in training, starting from the same initial condition. Even though such a strong generalization is considered, one-to-one correspondence between the two sequences is retained. Remember that this is accompanied by a strong compression in computation times of approximately three orders of magnitude on the same machine used to produce the training examples.

Due to the evidence for such strong generalization capabilities, we show as a last example what happens if the CRNN is used to generate sequences

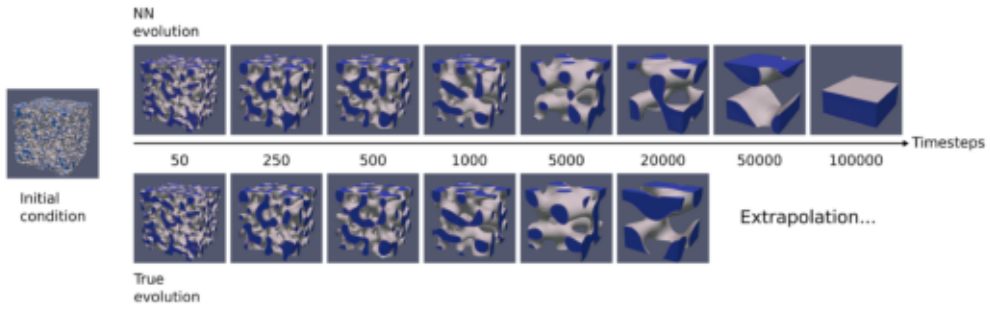

Figure 51: Extreme generalization test for the CRNN architecture. One-to-one correspondence is lost, but qualitative agreement is still present. The final configuration reached is consistent with the true stationary state.

that are *several orders of magnitude* longer than the original training one. In Figure 51, we report a NN simulation totaling 10^5 snapshots, which would correspond to 80 million time integration steps for the traditional finite difference scheme. For long times one-to-one correspondence between predicted and ground truth evolutions is lost. This is actually expected, as the CRNN accumulates prediction errors and the considered evolution is subject to bifurcations in the evolution: a slight delay in domain splitting between the two dynamics may propagate to potentially very different configurations. This is the same phenomenon observed and discussed for sharp interface evolutions in Figure 31. Still, the evolution stages provided by the CRNN are in line with the underlying physical process, at least on a qualitative level. Maybe the most striking characteristic is that the last stage reached by the NN prediction presents a flat interface between the two phases. This is indeed the stationary state which is expected for long enough domain coarsening if the initial and boundary conditions considered here are provided. Notice however that no such configuration has been observed by the CRNN during training, proving that the implementation is capable of extensive generalization.

Of course, comparisons are only qualitative, but promising. It would be interesting, therefore, to find a suitable indicator that is capable of providing a quantitative evaluation of the soundness of the evolution. Of course, due to the bifurcation phenomena discussed above, this cannot be a voxel-to-voxel distance (e.g. some L_p measure), as it would only imply one-to-one correspondence. In this respect, we are currently working on the possibility of analyzing scale behaviors or average quantities (e.g. the energy of domain partitions). Results will be collected in a future publication.

5 GANS: TACKLING STOCHASTIC DYNAMICS

In this last Chapter, we will discuss possible applications of Deep Learning approaches to stochastic systems. At variance with the rest of the Thesis, in this case a fully probabilistic approach is necessary. One of the most successful approaches developed in the last years will be employed, Generative Adversarial Networks (GAN).

The approach and the results that will be presented in this Chapter have been developed and obtained in collaboration with Professor Olivier Pierre-Louis during my stay in *ILM* at the *Université Claude-Bernard Lyon-I*. We are therefore hugely indebted to him as his experience in stochastic systems proved to be essential for what follows.

The Chapter is organized as follows. Section 5.1 introduces theoretical foundations for Generative Adversarial Networks. Section 5.2 shows how the original GAN formulation has to be adapted to a simple one-dimensional stochastic system. Specifically, motivations behind the noise injection procedure which is expected to be necessary to stabilize training for all lattice models are provided.

In Section 5.3 we briefly outline perspectives and applications to many interacting particle systems. This is currently one of the main research objectives we are pursuing.

Data on the single particle system are also published in Ref. [148].

5.1 GENERATIVE ADVERSARIAL NETWORKS

For once, we will start by introducing the Machine Learning method and then turn to the physical model. To tackle the training of probability distributions we will use Generative Adversarial Networks. The theory is very general and is based on the pioneering work of Goodfellow et al. [149].

Up to this point, we always assumed that Machine Learning model outputs had *some* probability distribution. Mainly, it was in the form of a conditional Gaussian probability density, whose mean was then parametrized utilizing a NN. This allowed us to introduce the Mean Squared Error Loss and convert the training procedure into a minimization task. In the end, however, the actual probabilistic nature of the learned model has never been considered. For instance, the variance of the Gaussian has never been specified. After all, we were interested in modeling *deterministic processes*, hence we can imagine that the probability distribution of the NN output is a degenerate, δ-like object. If we want to tackle a stochastic dynamical system, this is no longer an option.

The task of "learning a probability distribution" can be informally considered as finding a way to *extract or generate* new samples that are consistent with the training set. In many aspects, this approach surpasses the *super-*

vised training paradigm we used throughout the rest of the Thesis: at least for now, we will no longer consider a problem in which there are input and output variables. Instead, the training set is composed of generic data points, x which come from an unknown probability distribution.

Several, more traditional approaches allow for the learning of a probability distribution (or probability density). All of them, however, present some problems, mostly related to the calculation of the *partition function* [22] which leads to the requirement of performing Gibbs sampling to extract new samples (see for example Restricted Boltzmann Machines [150]). Here we will outline how Generative Adversarial Networks (GANs) represent a convenient alternative.

5.1.1 The generator network

As Neural Networks possess universal approximation capabilities, it seems there is no reason why they should not be able to approximate unknown, potentially very complex and multimodal probability distributions. As NN are differentiable models, we will discuss now on in terms of probability densities. Naively, one could search for an approximation for the true probability density $p_{data}(x)$ directly using a NN. As we discussed at the very beginning of Chapter 1, the main idea is that a good approximation should be "as close as possible" to the true object in terms of some similarity index. For probability distributions it is customary to consider the *Kullback-Leibler (KL) divergence* [151]:

$$D_{KL}(p_{data}(x)||\hat{p}_\vartheta(x)) = \int p_{data}(x) \log \frac{p_{data}(x)}{\hat{p}_\vartheta(x)} dx \tag{5.1}$$

where $\hat{p}_\vartheta(x)$ is the NN approximation defined by parameters ϑ. KL divergence has various nice properties. First, it vanishes if and only if the two probability (densities) are equal[1] and is positive in all other situations. Furthermore, it is strictly connected to information theory and information entropy [151, 22]. Notice, however, that it does not satisfy the properties of a distance (e.g. it is not symmetric), hence the name divergence.

Unfortunately, this approach has two main problems. First, even if optimization of Equation 5.1 is performed, having the probability density does not necessarily mean that sampling from the distribution is straightforward. Indeed, for high dimensional probability densities this is commonly done by some form of Gibbs sampling [21, 150], which is one of the problems we are trying to avoid. Second, and more importantly,*we have no direct access* to the true probability density, thus making it impossible to calculate exactly the log term.

Let us therefore consider an alternative approach. Sampling a random vector from high dimensional distributions is easy if its components are mutually independent, as this reduces to simple one-dimensional sampling. For example, extracting from a N dimensional Gaussian with a diagonal covari-

1 Technically, this holds on all Borel sets. For continuous probability distributions, their KL divergence is zero if and only if they are equal *almost everywhere*, i.e. except on measure zero sets.

ance matrix amounts to sample N Gaussian random variables. At the same time, NN universal approximation capabilities guarantee that it is possible, at least in principle, to approximate the mapping between some random vector z and a sample x coming from the data probability distribution. Notice that this is true as long as the vector space z is equal or larger in dimensionality with respect to the real data manifold, which in turn can be smaller than the dimensionality of data points (recall the concept of *manifold learning* and non-linear projections discussed in Section 1.7). The reader familiar with other ML approaches will recognize that this structure is very close to that of a Variational Autoencoder (VAE; see e.g. Ref. [152]), albeit only the decoder part of the Network is present, and normalizing flow models [153]. z is called a *latent variable* and the vector space where it is defined *latent space*, in continuity with the nomenclature used in these other models.

If the $z \to x$ mapping is learned, it is then easy to sample from the data distribution, as it only would amount to sampling from a known, simple distribution, such as a multidimensional Gaussian or uniform, and a forward pass through a NN. As the Network translating latent vectors into data space defines a generative model, we will call it *Generator network* and will indicate it with G.

We have still not described, however, an optimization procedure for the mapping: high probability density regions of the latent space should be mapped to corresponding regions in data space, *without explicit knowledge* of the true probability distribution. While seemingly desperate, it turns out that there is an elegant solution to this problem.

5.1.2 The adversarial approach

Goodfellow et al. [149] have proven that the problem can be solved, as long as an additional auxiliary, antagonistic network is introduced, which is called *Discriminator network*. Fundamentally, the Discriminator may be considered as a classifier whose task is to identify whether the data it is presented comes from the true distribution $p_{\text{data}}(x)$ or has been produced by the Generator $G(z)$, z being extracted from an arbitrarily chosen latent space distribution. The learning task is then converted into an *adversarial game* between the two competing networks, hence the name Generative Adversarial Network (GAN).

At least in the original formulation of Ref. [149], the Discriminator has an output $s \in (0, 1)$, which reflects the probability that data come from the true distribution rather than from the Generator: $s = 1$ means that the Discriminator is certain that its input comes from p_{data}, while $s = 0$ means that D is identifying data as coming from the Generator. The training of such a classification network is normally done by minimization of a loss function $\mathcal{L}_D(\vartheta_D)$ based on the *binary cross entropy* [149, 23, 25]:

$$\mathcal{L}_D(\vartheta_D) = -\frac{1}{2}\left(\mathbb{E}_{x\sim p_{\text{data}}}[\log(D(x|\vartheta_D))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)|\vartheta_G))]\right) \quad (5.2)$$

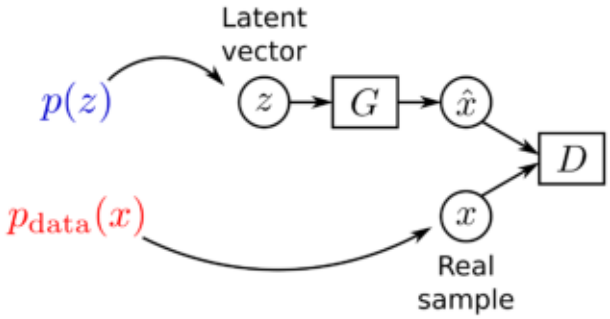

Figure 52: Sketch of the GAN approach. The latent vector z is sampled from a predefined distribution $p(z)$ and is mapped to $\hat{x}$ by the Generator G. Real samples are extracted from the true data distribution $p_{\text{data}}(x)$. The goal of the Generator is to fool the Discriminator D, while D tries to identify if samples come from G or the real data.

which can be derived by considering the KL divergence with a Bernoulli random variable. ϑ_D represents the set of parameters for the Discriminator network and ϑ_G are parameters of the Generator.

At the same time, the Generator is trying to fool the Discriminator, hence its objective is the maximization of the same Equation 5.2. As in Deep Learning loss functions are usually minimized, $\mathcal{L}_G$ is

$$\mathcal{L}_G(\vartheta_G) = \frac{1}{2}\left(\mathbb{E}_{x\sim p_{\text{data}}}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z|\vartheta_G)))]\right) \quad (5.3)$$

We now prove that the minimization of the objective functions indeed leads to a Generator that implicitly samples from the data distribution, provided that the involved NN capacity is high enough. We re-write Equation 5.2 in integral form:

$$\begin{aligned}\mathcal{L}_D[D] = &-\frac{1}{2}\left[\int p_{\text{data}}(x)\log(D(x))dx + \int p(z)\log(1 - D(G(z)))dz\right] = \\ &-\frac{1}{2}\left[\int p_{\text{data}}(x)\log(D(x))dx + \int p_G(x)\log(1 - D(x))dx\right]\end{aligned} \quad (5.4)$$

where we suppressed explicit dependencies from the NN parametrizations for ease of notation. In the second equality, we substituted integration with respect to the latent variable z using the implicitly defined probability density induced by the generator $p_G(x)$. For a fixed generator, minimization of Equation 5.4 implies the optimal Discriminator $D^*(x)$:

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} \quad (5.5)$$

as can be readily found by Euler-Lagrange equations [110].

Inserting back the optimal discriminator in the Generator loss Equation 5.3, we obtain:

$$\begin{aligned}\mathcal{L}_G[p_G] =& \frac{1}{2}\int p_{data}(x)\log\left(\frac{p_{data}(x)}{p_{data}(x)+p_G(x)}\right)dx + \\ & \frac{1}{2}\int p_G(x)\log\left(\frac{p_G(x)}{p_{data}(x)+p_G(x)}\right)dx\end{aligned} \tag{5.6}$$

where we rephrased everything as a functional of the implicit generator density p_G. Importantly, this quantity may be expressed in terms of the Jensen Shannon Divergence (JSD):

$$JSD(p||q) = D_{KL}\left(p\middle|\middle|\frac{p+q}{2}\right) + D_{KL}\left(q\middle|\middle|\frac{p+q}{2}\right) \tag{5.7}$$

which, as the name suggests, similarly to KL divergence measures a difference between probability densities and vanishes when $p = q$ almost everywhere[2]. In particular, the Generator loss function is given by:

$$\mathcal{L}_G[p_G] = JSD(p_G||p_{data}) - \log(2) \tag{5.8}$$

which has a global minimum value when the Generator probability density is equal (almost everywhere) to the true data one. Critically, we do *not* need at any time during training to actually access probability densities! This means that, at least in principle, if Discriminator training to optimality is alternated with a Generation loss minimization step, the overall adversarial game converges to a Nash equilibrium in which the Generator which samples from the data distribution, i.e. $p_G = p_{data}$. In this situation $JSD(p_G||p_{data}) = 0$, $\mathcal{L}_G = -\log(2)$, $\mathcal{L}_D = \log(2)$ and $D^*(x) = 1/2$, i.e. all input presented to the Discriminator are considered as equally likely to come either from the data or the Generator distribution. The same results may be derived minimizing Equation 5.6 with Euler-Lagrange equations with the constraint that p_G integrates to unity.

Unfortunately, GANs have proven particularly difficult to train in practice. Already from their invention in Ref. [149], some tricks have been proposed. First, training the Discriminator to optimality for every Generator minimization step is clearly impractical (or even impossible). For this reason, this is often substituted with a fixed number of $\mathcal{L}_D$ updates. Sometimes a single step is used. Indeed, this number of Discriminator iterations can be considered a hyperparameter in GAN training.

Another modification to the presented theory is that the Generator loss function is often substituted with the so-called *non-saturating* variant:

$$\mathcal{L}_G(\vartheta_G) = -\mathbb{E}[\log(D(G(z|\vartheta_G)))] \tag{5.9}$$

which can be verified to have the same equilibrium point and a minimum loss of $\mathcal{L}_G = \log(2)$. The motivation for this substitution comes from stronger gradients early in training [149]. Unfortunately, this is often not enough and

2 It turns out that we can proceed oppositely to what we have done here, *defining* a divergence (or other difference measure) between probability distribution and *derive* the corresponding adversarial game. See Ref. [154].

the reasons why training do not converge are very rarely clear. Indeed, the study of GAN training dynamics is at the moment a very active area of research [155, 156, 157].

One of the most dramatic failure modes is related to the so-called *mode dropping* phenomenon [158]. In brief, a Generative Adversarial Network is said to have encountered mode dropping if it extracts samples only from some modes of the data distribution, completely ignoring the others. Unfortunately, this kind of behavior is difficult to monitor and identify, to the point that one of the most reliable methods is still having a human looking at generated samples. These reasons make training Generative Adversarial Networks close to a "black art", in which trial end error is in the end often the only possibility to explore the effects of hyperparameters.

Despite these difficulties, GANs have many advantages. First, sampling from them is fast, especially if compared with methods that require a Gibbs sampling procedure, as it is sufficient to extract a new random z and perform a forward pass through the Generator Network. Second, they can leverage the flexibility of Neural Network approaches. In a few months after their proposal, a proliferation of different architecture emerged in the DL community [30, 159]. Importantly, convolutional GANs make it possible the efficient generation of image data (e.g. see Ref. [160] for the generation of photo-realistic images). Another useful possibility is the integration of *context* to the generated samples. If a *class label token* is added both in the Generator and the Discriminator inputs, it is possible to obtain a GAN that extracts conditional samples. This method is called conditional GAN (cGAN) [30] and will be useful in the following.

5.2 GANS FOR STOCHASTIC DYNAMICS

Now that we have outlined the main aspects of GANs, we will turn to how they can be applied to physical systems. Application of Generative approaches to statistical mechanics problems is somewhat natural, as they allow for accelerated or approximate sampling from complex distributions. Indeed, some works already explored the possibility of using GANs as samplers from configuration and compositional distribution [161, 162]. On the other hand, applications to stochastic dynamics are present but scarcer [163, 164].

Our core assumption is that a generative model can provide the next state of a stochastic system *provided* the current one. This does not require specialized Recurrent structures, as the Markov property 4.22 is assumed: if the next state of a Markov chain only depends on the present state, we can in principle use a GAN to model the individual state-to-state transition probabilities. Notice, however, that in this context the task is complicated by the fact that we are not assuming any prior knowledge of the shape of transition probabilities other than collected observations. One of the most critical aspects is that, at variance with common applications in text/image generation, the margin of error for GAN models in this context is very small, if ensemble properties are to be recovered. In the following, we will present a simple physical model that will constitute our test case and explain how

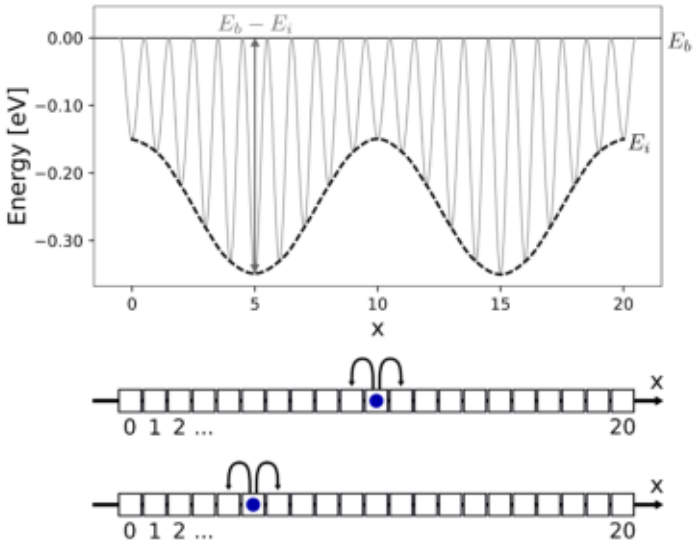

Figure 53: Skecth of the one-dimensional stochastic process considered. A particle diffuses on a one-dimensional lattice by either hopping to the left or the right. Hopping rates are defined by a position-dependent energy function

such a high level of accuracy can be obtained despite the difficulties in GAN training.

5.2.1 Physical model

We will consider a simple, one-dimensional diffusion problem on a lattice. The state of the system is described by the position of a particle x_t which moves at random under the influence of an external potential. A sketch of the process is provided in Figure 53.

The particle position cannot assume continuous values. Instead, only lattice lattice positions i can be visited. Motion is stochastic due to thermal fluctuations: for a particle at site i, the jumping rate is given by

$$r_i = \nu \exp[-(E_b - E_i)/k_B T] \tag{5.10}$$

where E_i is the (position dependent) energy of site i, E_b is the energy of the diffusion barrier, k_B is Boltzmann's constant and T is the temperature. As the problem is a toy system, there is no need to keep unnecessary constants: lengths will be measured in terms of lattice spacing so that x can be represented by integers $x \in [0, L]$. Specialized treatment is required for boundary points $x = 0$ and $x = L$: this time, instead of considering periodic boundary conditions, the particle is reflected and can only jump to the right (for $x = 0$) or left (for $x = L$).

For the energy function, a simple double well potential will be considered:

$$E_i = \frac{1}{2}(E_{max} + E_{min}) + \frac{1}{2}(E_{max} - E_{min}) \cos[4\pi i/L]/2 \tag{5.11}$$

where E_{max} and E_{min} are the maximum and minimum energy respectively and L is the domain length. In the following we will consider $E_{min} = -0.35$ eV, $E_{max} = -0.15$ eV and $L = 20$. In Figure 53 diffusion barriers are represented, as transition rates can be interpreted in terms of hopping between energy minima in transition state theory.

Clearly, we cannot assess the model quality through a one-to-one comparison between true and generated trajectories. We therefore need to resort to average quantities. In this respect, the simplest object is the probability distribution at thermodynamic equilibrium. Writing the master equation for the system and solving for the stationary state yields Boltzmann distribution at equilibrium [165]:

$$P_{eq,i} = \frac{1}{\mathcal{Z}} e^{-E_i/k_B T} \tag{5.12}$$

where $\mathcal{Z} = \sum_{i=1,L} \exp[-E_i/k_B T]$ is the partition function. For the present application, the temperature will be fixed: $T = 500$K.

The goal of the GAN approach we are discussing here is not only to calculate equilibrium properties. Indeed, if that was the only scope, it would probably have been more effective to collect samples directly from the Boltzmann distribution and check that the obtained GAN statistics are consistent with Equation 5.12. Fortunately, due to the system's simplicity, it is also possible to calculate the average time required for a particle in one minimum of the energy function $x = 5$ or $x = 15$ to reach the other. In the continuum limit $L \gg 1$ [165],

$$\tau = \frac{Ld}{2} e^{-\beta_+} I_0[\beta_-], \tag{5.13}$$

where $\beta_\pm = (E_{max} \pm E_{min})/2k_B T$, $d = 10$ is the distance between the minima of the potential, and I_0 is the modified Bessel function of the first kind.

Trajectories from stochastic process defined by Equation 5.10 and the other parameters provided can be simply sampled using simple Kinetic Monte Carlo (KMC) simulations. This allows for fast construction of the training set, which is composed of 6×10^5 couples $(x_t, x_{t+\Delta t})$ coming from 1200 independent trajectories. Here x_t is the particle position at time t and $x_{t+\Delta t}$ the position at time $t + \Delta t$.

The choice of a fixed time interval is very different from what is typically observed in standard KMC methods: Kinetic Monte Carlo approaches typically simulate the (stochastic) *next state of the system* in the Markov chain, irrespective of the physical time required to reach it. This is a double-edged sword, however. On one side, this aspect allows one to skip long time intervals if rates are all very small, de facto speeding up simulations with respect to many alternative methods. On the other hand, when very high rates are present, the Markov chain is bounded to pass through low residence time configurations, possibly employing a huge number of steps before phenomena of interest happen, in the so-called *low-barrier problem* [166]. Choosing a fixed Δt absorbs high-rate phenomena in transition probabilities.

The time interval was chosen to be $\Delta t = 10^3$ in the units of $\exp[E_b/k_B T]/\nu$. This is an intermediate value with respect to mean residence times: the average time interval a particle stays in maximum energy lattice points is

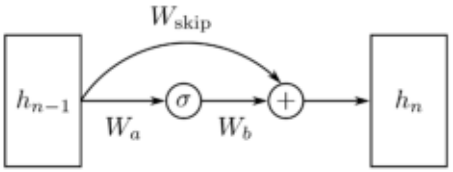

Figure 54: Sketch representing a ResNet module. W_a, W_b and W_{skip} represent affine transformations.

approximately $\Delta t/30$, while for maximum energy positions, this is $3\Delta t$. This choice is arbitrary, but remember that too small values of Δt would make the generation of reasonable trajectories to calculate stationary properties too long, while too large values would cause a collapse of the GAN procedure back to extracting directly from the equilibrium distribution: in the limit of infinite time, the conditional probability $P(x_{t+\Delta t}|x_t)$ simply yields the Boltzmann distribution 5.12, as should be expected from the Markov chain formulation of the process.

5.2.2 ResNet architecture

For the Generator and Discriminator models, we will consider ResNet architecture [147]. The main idea behind this class of NN is that instead of simply stacking layers on top of each other as in standard fully-connected architectures, the $n-th$ hidden layer value is obtained as

$$h_n = W_b \sigma(W_a h_{n-1}) + W_{\text{skip}} h_{n-1} \tag{5.14}$$

where W_a and W_b are usual affine transformations (or convolutions) and σ a suitable non-linear activation. As W_{skip} directly connects h_n and h_{n+1} is called a *skip connection*. A sketch of such a computational module is shown in Figure 54.

One of the main advantages of ResNet architectures is that skip connection allows one to enhance the gradient connecting distant hidden layers, due to the presence of an affine transformation mapping between all h_n (remember that the composition of affine maps is itself an affine map). This serves to mitigate vanishing gradient problems, which, although less severe than in RNN, are also present in very deep fully-connected structures, especially if squashing functions such as hyperbolic tangents or sigmoids are used [22]. This peculiar structure not only presents advantages in "gradient flow", but also has the possibility of exploiting an analogous result of Proposition 1: at variance with simple fully-connected NN, increasing the depth of ResNets always results in an extension of the representable functions (although this says nothing about resistance to overfitting).

5.2.3 Generator and discriminator models

The Generator approximates the mapping of the present state x_t to the next state $x_{t+\Delta t}$ in a conditional GAN fashion, which complies with the markovian nature of the model considered:

$$\tilde{x}_{t+\Delta t} = G(x_t, z) \tag{5.15}$$

where z is the latent variable and x_t is the previous state of the system. In practice, this simply means that the x_t is concatenated to the latent vector z in G input layer. The choice of the dimensionality of the latent space can again be regarded as a hyperparameter. As long as it is higher than that of the real data *manifold* (not necessarily the whole data space), this number should be unimportant. Once the generator has been trained, it is possible to generate arbitrarily long stochastic series by iterative application of G, in a recurrent fashion. For example:

$$\tilde{x}_{t+2\Delta t} = G(\tilde{x}_{t+\Delta t}, z') = G(G(\tilde{x}_t, z), z') \tag{5.16}$$

where the different notations z and z' make it explicit that a new random latent variable is extracted at every forward run of the Generator. Notice that, similar to what we have seen in output recurrence (see Section 4.2), there is a critical difference between training and sequence generation: in the first case, the input comes from the training set (no tilde on top of the x), while in the latter G input has been itself generated from the Network.

The Discriminator is simpler: given the input $(x_{t+\Delta t}, x_t)$ or $(\tilde{x}_{t+\Delta t}, x_t)$, it should output a number in $(0, 1)$ representing the confidence level that pairs come from the dataset and not from the Generator. As is standard for binary classificator, this can be easily done by applying a logistic sigmoid to the output layer of D.

A total of 7 ResNet modules have been used both for the Generator and the Discriminator Network.

5.2.4 Training and Measures

In principle, we are now ready to train our first GAN. Figure 55 shows the resulting training plot for the procedure proposed by the original GAN paper [149] using the non-saturating variant of the Generator loss and Adam optimizer with the same parameters suggested in the paper (remember that in this case losses should converge to $\log(2)$ and not zero). Clearly, something is going wrong: loss functions exhibit wild oscillations before settling to high values (for G) or vanishing (for D). The failure of the training procedure can also be identified by observing the comparison between a generated trajectory and a sample of a true trajectory on the right panel.

The reason behind the training non-convergence is somewhat subtle and is related to implicit assumptions in Section 5.1.2. In particular, everything we said was based on the tacit assumption that random variables are *absolutely continuous*, i.e. they have non-degenerate probability densities. For the simple particle diffusion on a one-dimensional *lattice*, however, this is not

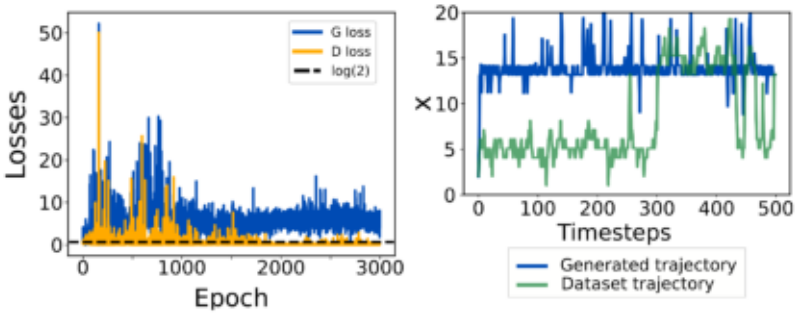

Figure 55: Training result via direct implementation of the original GAN formulation for the one-dimensional particle diffusion problem (left) and an example of generated trajectory compared with a training set one (right).

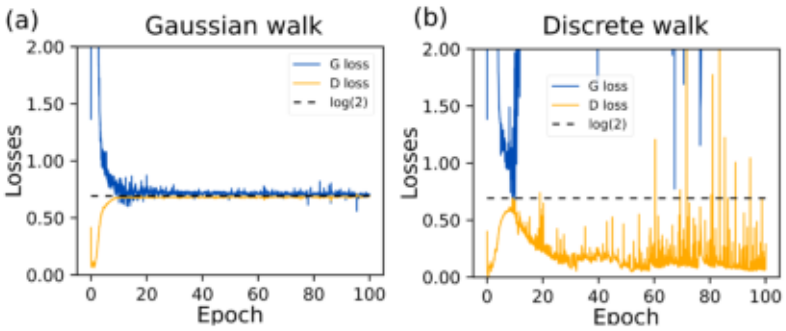

Figure 56: Comparison between the GAN training procedure for a continuous Gaussian random walk (b) and a random walk on a lattice (b) in 1D. As can be observed, the continuous case readily converges, while the discrete does not.

true. Indeed, the fact that x can only take integer values means that its distribution is different from zero only at integer numbers in $[0, 20]$. Formally, $p(x_{t+\Delta t}|x_t)$ has a *support* [3] with zero measure. The fact that the set of integers $\mathbb{Z}$ has a null measure on the real line $\mathbb{R}$ is one of the basic results in measure theory. As a further confirmation that this is the origin of the problem, we show here the comparison between the same training procedure performed on data coming from an unbiased Gaussian walk (Figure 56(a)) and an unbiased random walk on a lattice (Figure 56(b)). It is evident that the discrete walk yields a divergent behavior, while the Gaussian process quickly settles to the theoretical $\log(2)$ value.

We stress the fact that such a training instability problem is not only a pathological situation arising from our bad choice of stochastic system: if the manifold hypothesis is true, we should expect *almost all probability distributions* to have zero measure in the *data space*. Additionally, lattice models represent a huge class of tools in computational Materials Science, hence it is essential to devise a strategy that makes it possible to train on these too.

A more general study of the instability phenomenon has been performed by Arjovsky and Bottou [155]. We will not repeat here their analyses, as they are heavily based on mathematical topology and measure theory. The overall result, however, may be stated as follows: every time the Generator and

3 Actually, being x a discrete random variable, it should not even have a probability density in a strict sense.

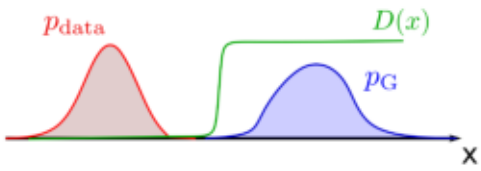

Figure 57: Illustration of the vanishing problem which may arise if data and generator distributions have zero measure overlap. Since D is constant over the support of both distributions, but with different values, it can perfectly identify the two distributions without providing useful training gradients

the real data distributions intersection has zero measure, the original GAN training procedure will either be vanishingly slow or unstable, depending on whether the original Generator loss function 5.3 or the non-vanishing alternative 5.9 are used. While we used the latter and indeed observed unstable behavior, the first case is easier to illustrate with a simple sketch in 1D, reported in Figure 57. If the Generator and true data probability distributions have non-intersecting support, it is possible to have a *perfect Discriminator* which is constant (or almost constant) in both regions. This makes training the Generator impossible, as the derivative of D with respect to the input, which in turn is necessary to calculate the gradient of $\mathcal{L}_G$, vanishes almost everywhere. This is one of the reasons why training D for many more optimization steps than G often results in diminishing returns.

Luckily, the same Ref. [155] provides some possible solutions for this problem. For instance, if it is possible to lift the degeneracy of probability distributions, then we can recover all nice convergence results discussed in Section 5.1.2. This can be done straightforwardly by simply *adding* a zero-mean normal random variable ϵ to the input of *both* the Generator and the Discriminator[4]. As the Gaussian function support is not bounded, in principle this should work for arbitrary standard deviation σ.

In practice, however, a too-large σ hides the lattice-like nature of the physical problem we are considering, making the GAN converging but strictly less useful. On the opposite side of the spectrum, if σ is too small, the stabilization benefit is almost non-present and the training does not converge. In the main text and appendices of Ref. [148], we report an in-depth analysis of the effect of σ showing that the optimal is 0.25 in lattice spacing units. This value can be related to the physical system considered, as it allows for sufficient overlap between Gaussians centered at adjacent lattice points while retaining the possibility of actually identifying individual positions.

The loss plot obtained with the above procedure is shown in Figure 58. Importantly, the stabilization brought by noise addition allows for the use of more training steps for D safely. Specifically, 20 minimization steps for D have been performed for every G minimization step. The interested reader can find pseudocode for the training loop in Appendix E. As it can be seen, both the loss functions get close the the ideal $\log(2)$ value, despite some persistent oscillations. Similarly, the generated trajectory in Figure 58 is qualitatively similar to the one obtained by KMC.

4 In this section there is no elasticity involved. The reader should therefore not worry about possible ambiguities with stress/strain fields

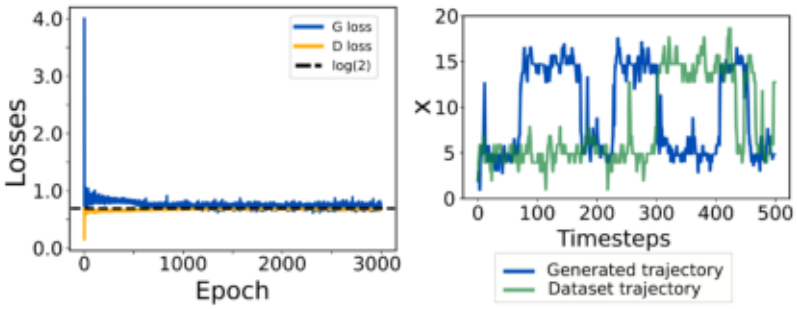

Figure 58: Training result when noise is added to the Network inputs (left) and an example of generated trajectory compared with a sample from the training set (right).

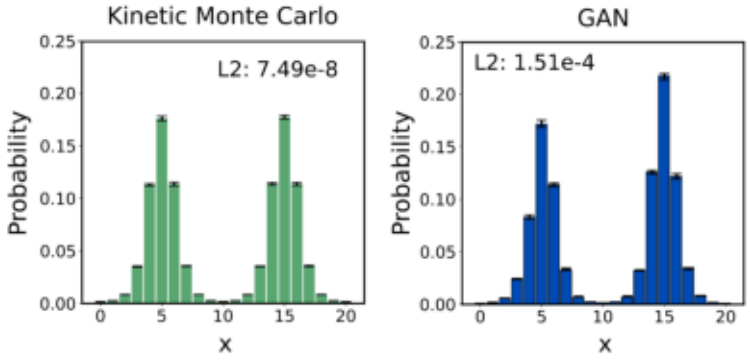

Figure 59: Comparison of the equilibrium state distribution for KMC (left) and a single GAN model at the end of training (right). L_2 distance with the theoretical distribution is reported in the insets.

Before moving on, we should add some comments on how the noise introduction modifies sequence generation. As we have been already very technical, we will just say here that new Gaussian noise is extracted and added to G input whenever the next state of the system is generated. There is however no need to add ε to G outputs when they are saved. Appendix F discusses the topic in more detail.

5.2.5 A first look at results

Now that the adversarial game procedure is convergent, let us comment on the trained models' performances. First, we will discuss stationary state properties. As the system is ergodic, the stationary state distribution may be obtained through the histogram of positions for a long enough trajectory. Specifically, Figure 59 shows such quantity as obtained from 5×10^5 steps iterative applications of G at the end of training as compared with the distribution obtained by KMC simulations. Error bars have been estimated considering 10 independent such trajectories. The difference between the probability distributions is noticeable, as the one obtained by G is evidently asymmetric. The strong deviation can further be confirmed on quantitative grounds if the L_2 difference with the theoretical Boltzmann distribution 5.12 is considered (reported in inset), which is more than 3 orders of magnitude higher for GAN.

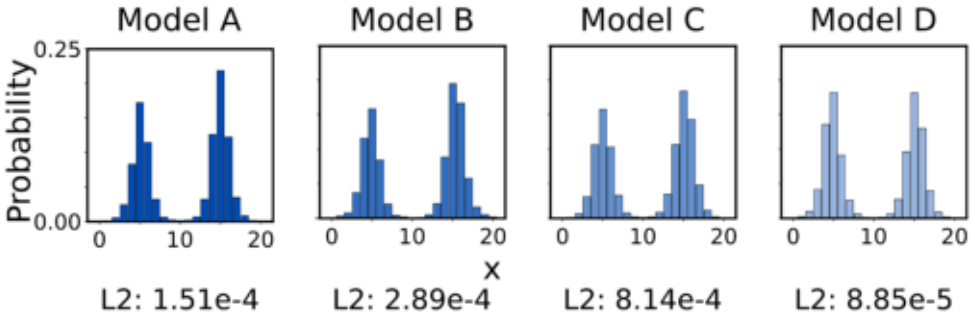

Figure 60: Comparison between the equilibrium distributions of 4 models 50 epochs apart at the end of GAN convergent training. L_2 deviations with the theoretical distribution are reported below each histogram.

The inconsistency between KMC and GAN distributions may be explained by the effects that slight differences in transition probabilities have on the infinite time limit of the stationary state distribution. This can be appreciated if different Generators at the end of the training procedure, i.e. when both losses oscillate around the Nash $\log(2)$ value, are considered. The same procedure of Figure 59 is reported for 4 such models 50 epochs apart in Figure 60, together with the L_2 distance from Boltzmann distribution. Even though oscillations in $\mathcal{L}_G$ and $\mathcal{L}_D$ are small, this is sufficient to create dramatic differences.

It seems that we are at an endpoint and GANs cannot be used for learning this class of models at a high level of accuracy, despite the use of the specialized method which causes training stabilization. We still have one ace up our sleeve, however, and it turns out that it is possible to exploit model oscillations to our advantage.

5.2.6 GANs multi-model ensemble

In Section 4.4.1, we discussed the possibility of creating NN ensembles to obtain a stronger model through averaging, a procedure called *bootstrap aggregation*. Crucially, the more model predictions are different, the more the method is effective. Although we are not using bootstrap samples, it seems that this is the perfect premise for the problem we are facing here: at the end of each epoch, Generators produce very different samples but, although individually "reasonable", none of them is good enough to provide the accuracy required. The idea of averaging NN models obtained during a single training has already been explored [167].

In the present application, however, we are faced with an additional problem: by construction, we cannot easily access the conditional probability $p_G(x_{t+\Delta t}|x_t)$. The whole GAN idea revolved around the possibility of bypassing this quantity all along. In order to solve the problem, we introduce a modified stochastic process to generate trajectories for GAN models. Instead of selecting an individual Generator, we retain N different ones, G_i, selected in the region in which $\mathcal{L}_G$ oscillates around the convergence value, e.g. the last N training epochs. At every step, one of the models is selected

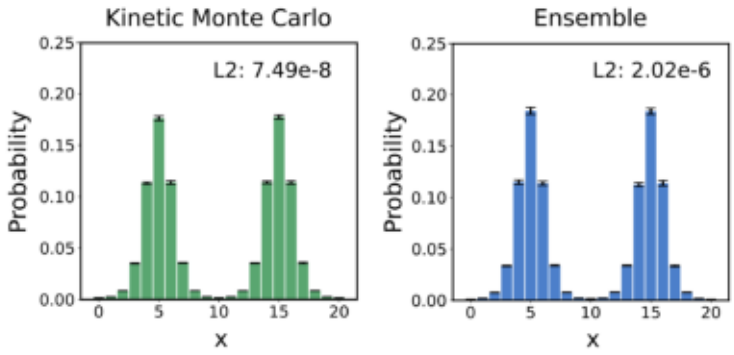

Figure 61: Caption

uniformly at random and the subsequent state of the system is generated through the usual procedure:

$$\tilde{x}_{t+\Delta t} = G_i(\tilde{x}_t, z) \tag{5.17}$$

Calculating the conditional probability $p(\tilde{x}_{t+\Delta t}|\tilde{x}_t)$ for the next state of the system for the above-described stochastic process is straightforward and just requires marginalizing on the model used for individual steps:

$$p(\tilde{x}_{t+\Delta t}|\tilde{x}_t) = \sum_i p_{G,i}(\tilde{x}_{t+\Delta t}|\tilde{x}_t) p_i \tag{5.18}$$

where p_i is the probability of selecting Generator i and $p_{G,i}(\tilde{x}_{t+\Delta t}|\tilde{x}_t)$ is the (implicitly defined) conditional probability from which G_i draws samples. Remembering that Generator models are selected uniformly

$$p_i = \frac{1}{N} \implies p(\tilde{x}_{t+\Delta t}|\tilde{x}_t) = \frac{1}{N} \sum_i p_{G,i}(\tilde{x}_{t+\Delta t}|\tilde{x}_t) \tag{5.19}$$

which is the desired average of Generator distributions. Importantly, at no point of this multi-model procedure we are required to explicitly access the probability density $p_{G,i}$ and this does not imply any additional computational burden during training, as we are exploiting models which have been anyway produced by the optimization algorithm. Of course, this approach introduces new hyperparameters, i.e. the number of models to be used in the ensemble and how to select them. Some information on the effect of this choice can be found in Appendix E of Ref. [148].

Armed with this *multi-model* Generator procedure, we can now try again to extract stationary and kinetic processes for the single particle diffusion process. Figure 61 reports the equilibrium distribution obtained with the same procedure discussed in Section 5.2.5 using as an ensemble the Generators at the end of the last 300 training epochs (KMC distribution is reported again for reference). This represents a big improvement with respect to Figure 59, allowing for quantitative predictions: not only the distribution is now far more symmetric, but L_2 distance with Boltzmann distribution decreased by 2 orders of magnitudes.

With this result, now it also makes sense to compare kinetic properties. If many different sequences are initialized with a particle in one minimum

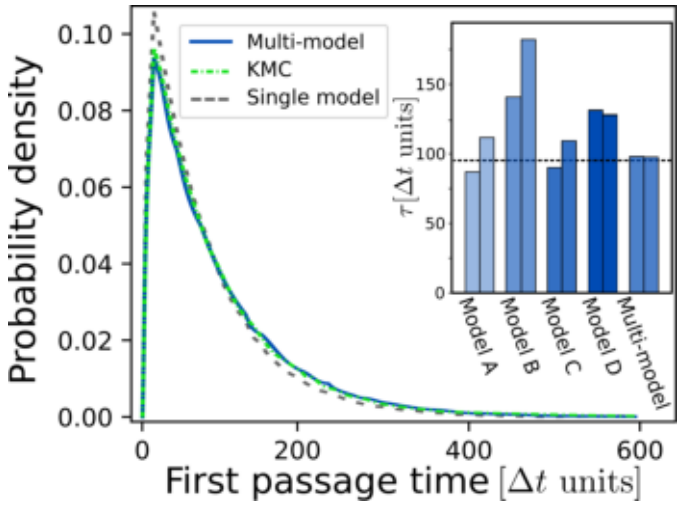

Figure 62: First passage times analysis for the multi-model GAN ensemble (blue line), KMC simulations (dashed green) and a single model (dashed gray). In inset, mean first passage times are reported for individual models and the GAn ensemble. The dashed black line represents the analytical prediction. The two bars correspond to left-to-right and right-to-left passage times respectively.

and the number of Δt intervals required to reach (or surpass) the other is considered, it is possible to extract from trajectories the first passage time τ. Using 10^5 independent runs, this quantity has been estimated using KMC, the GAN multi-model approach and the individual models of Figure 59. Figure 62 reports the probability distribution of τ obtained from 10^5 independent trajectories for each of these methods. As can be seen, the ensemble result is almost in one-to-one correspondence with the KMC curve, while the individual model overestimates the probability of smaller τ values. A further comparison can be done in terms of average first passage times. Using Equation 5.13, the KMC-estimated value presents negligible deviations from the analytical prediction (95.2 vs 9.14 respectively, in units of Δt). The barplot in the inset to Figure 62 reports this quantity for the four models of Figure 60 and the ensemble (analytical prediction is represented with the dashed line). Left-to-right and right-to-left passage times are reported independently to highlight asymmetries. The multi-model GAN approach is the only one that is capable of providing quantitatively accurate predictions, with an average τ only 3% off the analytical value.

5.3 TOWARDS COMPLEX SYSTEMS

Results in the previous Section are promising for possible applications of GANs in computational statistical mechanics. While interesting from a methodological point of view, the simple 1D case is mostly a proof of concept: there is no real computational advantage (actually KMC simulations are faster) in applying this method to simple 1D problems. If applications to more complex models are proven, this may change.

Figure 63: Example of a KMC simulation results for the evolution of a simple stripe under edge diffusion. As time progresses, the roughness of the interface increases.

5.3.1 KMC model for surface diffusion

Surface diffusion processes involving thermal fluctuations can be conveniently simulated by a lattice Kinetic Monte Carlo approach. For simplicity we will consider a 2D square grid, in which each site can be either occupied or unoccupied by a particle, effectively implementing the same model as Ref. [168] but disregarding the coupling with the concentration field, hence conserving the total number of particles in the system. The stochastic process is explicitly Markovian and at each KMC step a particle at random hops to a first or second neighbor site. The jump rate is given by a simple exponential Arrhenius form, analogous to Equation 5.10:

$$r_i = \nu \exp\left[\frac{nJ - E_b}{k_B T}\right] \tag{5.20}$$

where n is the number of nearest neighbors, $J < 0$ is a bonding interaction term, E_b is a diffusion barrier, k_B Boltzmann constant and T is temperature. To enforce so-called *edge diffusion* conditions [32], which results in particles only moving along domain boundaries, we will impose that the particle cannot diffuse to a lattice position if it is already occupied or if this means having zero first or second neighbors. If simple, straight stripe-like domains are used as initial conditions, at finite temperature the system exhibits a progressive increase of the free surface roughness W^2 (see Figure 63) [169]. In 2D the domain free surface can be described in the same fashion as in Chapter 3 with a profile function $h(x)$, as long as thermal fluctuations do not introduce overhangs in the free surface geometry. The roughness at time t may then be defined as the variance of $h(x,t)$:

$$W^2(t) = \frac{1}{L}\int (\overline{h}(t) - h(x,t))^2 dx \tag{5.21}$$

where $\overline{h}(t)$ is the (time dependent) average profile height and L is the domain length in the x direction.

As for the single-particle system, it does not make sense to perform a one-to-one comparison of stochastic trajectories. Instead, we have to resort to average quantities. Similarly to what was discussed in Section 5.2.1, we therefore identified two different quantities related to equilibrium and ki-

netic properties respectively. At thermodynamic equilibrium, the stationary state roughness is given in the continuum limit as:

$$W_{eq}^2 = \frac{Lk_BT}{12\tilde{\gamma}} \tag{5.22}$$

where $\tilde{\gamma}$ is the surface stiffness. For the problem at hand, this quantity may be related to the interaction term through [169]:

$$\tilde{\gamma} = \frac{k_BT}{2\sqrt{a}} \sinh^2\left[\exp\left(\frac{-J}{4k_BT}\right)\right] \tag{5.23}$$

where a is the lattice size.

For kinetic properties, things are more complex. For small times, it is a well-known result [170] that in the edge diffusion case considered:

$$W^2(t) = \frac{1}{\pi}\Gamma\left(\frac{3}{4}\right)\left(\frac{2\tilde{\gamma}Mt}{k_BT}\right)^{\frac{1}{4}} \tag{5.24}$$

where M is the mobility constant for diffusion and Γ is the Gamma function. Notice that the result is irrespective of the domain size L, as should be expected for early stages in the dynamics. Intermediate behaviors can also be analyzed, but the formalism is heavier and will be omitted here, especially as GAN results are still preliminary.

5.3.2 GAN model adaptation and preliminary results

To construct a suitable training set for GAN training, a total of 300 independent simulations on a 128×128 grid with the same initial stripe configuration have been performed and a total of 1000 snapshots equally spaced in time representing the lattice configuration have been collected. The time interval Δt between subsequent states was set to 5×10^5 in terms of $\nu \exp(-E_b/k_BT)$. Other fixed quantities are $T = 290$ K and $J = -0.1$ eV. Due to the similarity with the Phase Field representations of Section 4.1.4, we will indicate the binary grid defining the state of the system by φ.

While this 2D, many-particle system yields some similarities with the simple one-dimensional diffusion case, there are however important differences. First and foremost, the spatial nature of input-output data suggests the use of a convolutional structure. Second, the number of particles has to be conserved during the evolution. Third, if we consider the state of the system as a 128^2D vector, it is easy to observe that we still encounter the null measure problem discussed in Section 5.2.4. The combination of these effects, plus possibly some others that we have not identified yet, and the delicate convergence properties of GANs clearly makes this training task much more complex than the simple 1D example, to the point that a completely satisfactory solution has been elusive to us up to this point. As a reference, we show in Figure 64 the best loss plot produced so far for the parameters provided.

We will now discuss what are the actions we considered to adapt the strategy for the 1D problem to this class of surface diffusion simulations. Regarding the use of convolution operations, the solution is straightforward: it is sufficient to substitute generic affine mappings with convolutions. This

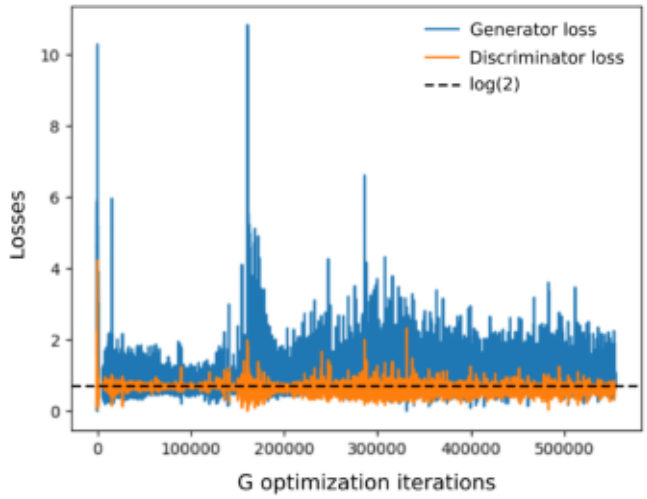

Figure 64: Loss plot for GAN training for the many particles edge diffusion case. While no divergent behavior is observed, strong oscillations affect the final result.

would also allow the generalization to domains of arbitrary size. Some additional care should be considered, however, as there is a subtle interplay between the choice of Δt and the locality assumption. In principle, KMC simulation methods are *global* as the system advances in time through the Markov chain of states *as a whole*. On the other hand, CNNs assume that what happens at a given pixel is only influenced by its surroundings. Indeed, if Δt is so small that only one Monte Carlo move happens (in our case exactly one pixel "turns off" and a single pixel "turns on"), then the assumption of locality is not satisfied. Similarly, if $\Delta t \to \infty$, the next state of the system is extracted from the equilibrium distribution (if one exists), which is again a global property. For intermediate Δt, however, the state of the system $\varphi_{t+\Delta t}$ is a *local* quantity. From one side, particle motion by diffusion is limited by an activation barrier, implying that there is a finite mean distance a particle may have moved in Δt. On the other hand, if there are many individual particle jumps, the NN does not have to identify on a global level which ones hopped and which have not. Under the assumption that the previous argument holds for the training set we are considering here, we therefore updated the ResNet architecture described in Section 5.2.3 with convolution operations. z itself is also promoted to a multi-channel image with the same resolution of φ (in the following, 75 channels have been considered, specifically).

To respect the conservation of the number of particles, in principle it could be possible to enforce this condition approximately with penalty terms in the loss function, as discussed in Section 4.3. Notice that in this case, however, it may result in difficult-to-control interactions with the adversarial game. This is particularly critical, since the convergence properties of GAN are delicate, as we already discussed in this Chapter. Hence, a similar strategy to the one discussed in Section 4.5.1 has been considered. The next stage of the system $\hat{\varphi}_{t+\Delta t}$ is calculated as

$$\hat{\varphi}_{t+\Delta t} = \varphi_t + \Delta\varphi_t = \varphi_t + G(\varphi_t, z) \tag{5.25}$$

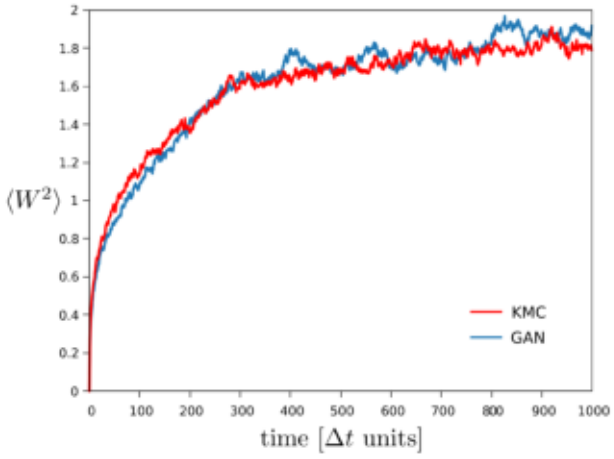

Figure 65: Roughness comparison as obtained from KMC simulations (red line) and GAN dynamics (blue line). Time is reported in Δt units and roughness in lattice spacing.

using a conditional GAN to predict the (stochastic) variation $\Delta\varphi_t$, which is enforced by construction to have zero spatial integral. This can be done by simply removing the average output of the Generator network in the last layer.

Lastly, as we have anticipated, we have to take care of the null measure problem already discussed. In this case, the binary nature of lattice sites is not helping. If φ is interpreted as a L^2D vector, L being the side of the square simulation cell, the KMC describes a random walk on a discrete, although huge, set in $\mathbb{R}^{L^2}$. This means that the noise injection procedure proposed in Ref. [155] is critical if stable training is to be achieved, at least if the original GAN formulation is used. On this side, there are at least two possible ways to "smear out" the data and generator distribution. A first possibility is again the addition of normally distributed random noise ε at each pixel in the image. Notice, however, that φ pixels take in the training set the values of either 0 or 1 *with no possible intermediate value*. If we use the same rule of thumb derived for the 1D system in which $\sigma \approx \delta/4$, being δ the spacing between available values, this leads easily to phase field representation with φ unphysical values. Additionally, as noise is not spatially correlated, this resulted in generated states heavily affected by artifacts in our tests. Maybe a promising option is represented by some kind of spatial smoothing. This allows us to fill in intermediate values between 0 and 1 near boundary regions, which in turn makes it possible to reduce the value of the noise standard deviation σ. Tests on the interplay between these strategies are still ongoing.

As a preliminary result, we show the comparison between the average roughness $\langle W^2 \rangle(t)$ obtained by GAN predictions and KMC simulations (Figure 65). The multi-model procedure of Section 5.2 has been used, as it proves to be extremely beneficial. Although the similarities are striking, notice that this result was obtained using an aggressive spatial smoothing approach, corresponding to bilinear interpolation in image resizing. Additionally, stable convergence to the level discussed in the Section 5.2 has still not been achieved, as was shown by the loss plot in Figure 64. Alternative GAN formulations, such as Wasserstein GANs [42] are also being considered.

We hope that in the following months, solutions to these issues can be found, as this would enable the use of GAN methods to tackle complex, high-dimensional stochastic problems, effectively providing a new computational tool for Materials Science.

CONCLUSIONS

In this Thesis the possibility of applying some of the modern Machine Learning tools to computational Materials Science has been explored. This required the synergy between statistical methods, computer science, mathematics, and physical considerations. We showed some applications, proceeding from the simpler to the more sophisticated ones. Sometimes, direct application of ML methods as they were originally developed in "pure" big data context was more or less straightforward, as in the case of dislocation interaction approximation (Chapter 2). In other cases, the analysis of the problem at hand also showed how some specialized structures emerge naturally, as in the case of Convolutional (Chapter 3) and Recurrent Neural Networks (Chapter 4). Other times, the interplay was more subtle and required careful examination of the underlying models, as in the GAN case of Chapter 5.

In most of this Thesis, we focused on prototypical cases, searching for trends and strategies that could be applied to more general situations. In the future, more complex physical problems will be tackled. All of the methods presented in this Thesis are open to extensions: due to the flexibility, approximation capabilities, and modularity of NN architectures, they can be generalized in several ways. Since we already discussed some of these possibilities in the main text at length, we only report here what are in our opinion the most promising directions.

The construction of a unified framework, possibly in three dimensions, between results in Chapter 2 and 3 is certainly intriguing: leveraging the NN architecture described in this Thesis, it could be possible to fully tackle the evolution of strained films involving both *elastic* and *plastic* relaxation mechanisms on "human" scales. This has important real-world applications, since dislocation and morphology control are essential for many technological steps in the semiconductor [58, 59] industry for example, and trial and error experimental searches are often too expensive. One of the main problems to be solved remains the description of three-dimensional dislocation configurations.

Convolutional Recurrent Neural Networks presented in Chapter 4 are probably the most straightforward approach to generalize to other phenomena. If multiple input channels are provided, applications to multi-physics simulations are possible. In this respect, our research group has already started to study the possibility of integrating elastic terms in the microstructural evolution of materials. Another feature that was not exploited in this Thesis is the possibility of using CRNN for sequence classification: spatiotemporal sequences can be in principle used as the input for a Network that learns how to extract additional, unknown parameters, such as the temperature or the load conditions driving a given morphological evolution. Another opportunity that we mentioned at the beginning of the Thesis is also the application of the same methods to experimental data directly: as

NNs are agnostic on the origin of training examples, there is no reason why CRNN should not work on sequences of experimental micrographs, for example. The main limitation here is the construction of a suitable training set.

The most ambitious extension is probably the one regarding the application of GANs to stochastic dynamics. As discussed at the end of Chapter 5, many difficulties must be solved. Preliminary results, however, are promising. If training difficulties are surpassed, this could allow a new computational approach encoding traditional KMC lattice models but alleviate the low-barrier problem.

ML methods are flexible, but the precision requirements for Materials Science applications are high when compared to other big data approaches. If this class of objects is to be a new tool for computational material scientists, their in-depth understanding will be necessary.

LIST OF PUBLICATIONS

- **Lanzoni, D.**, Rovaris, F., Montalenti, F. (2020). *Computational analysis of low-energy dislocation configurations in graded layers.* Crystals, 10(8), 1-23 [10.3390/cryst10080661].
- **Lanzoni, D.**, Rovaris, F., Montalenti, F. (2022). *Machine learning potential for interacting dislocations in the presence of free surfaces.* Scientific Reports, 12(1) [10.1038/s41598-022-07585-7].
- **Lanzoni, D.**, Albani, M., Bergamaschini, R., Montalenti, F. (2022). *Morphological evolution via surface diffusion learned by convolutional, recurrent neural networks: Extrapolation and prediction uncertainty.* Physical Review Materials, 6(10) [10.1103/PhysRevMaterials.6.103801].
- **Lanzoni, D.**, Pierre-Louis, O., Montalenti, F. (2023). *Accurate generation of stochastic dynamics based on multi-model generative adversarial networks.* The Journal of Chemical Physics, 159(14) [10.1063/5.0170307].
- Ge, G., Rovaris, F., **Lanzoni, D.**, Barbisan, L., Tang, X., Miglio, L., et al. (2024). *Silicon phase transitions in nanoindentation: Advanced molecular dynamics simulations with machine learning phase recognition.* Acta Materialia, 263(15 January 2024) [10.1016/j.actamat.2023.119465].
- Martín-Encinar, L., **Lanzoni, D.**, Fantasia, A., Rovaris, F., Bergamaschini, R. and Montalenti, F, *In preparation*

A ADAM OPTIMIZATION

We report the Adam optimization algorithm [40] for completeness. The values β_1 and β_2, controlling momentum and gradient rescaling respectively are to be considered as hyperparameters.

$lr \leftarrow$ learning rate ;
$\beta_1, \beta_2 \leftarrow$ set beta parameters ;
$m_0 \leftarrow$ initialize momentum ;
$v_0 \leftarrow$ initialize second moment ;
$N_{steps} \leftarrow$ number of iterations ;
$\varepsilon \leftarrow$ division stability parameter ;
$\mathcal{L} \leftarrow$ loss function ;
for $t = 1{:}N_{steps}$ **do**
  $g_t \leftarrow -\vec{\nabla}_\vartheta \mathcal{L}(\vartheta_{t-1})$;
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$;
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)^2 g_t^2$;
  $\hat{m}_t \leftarrow m_t / (1-\beta_1^t)$;
  $\hat{v}_t \leftarrow v_t / (1-\beta_2^t)$;
  $\vartheta_t \leftarrow \vartheta_{t-1} - lr \hat{m}_t / (\sqrt{\hat{v}_t} - \varepsilon)$;
end

B | LINEAR ELASTICITY FUNDAMENTALS

A big part of this Thesis work is dedicated to mesoscale behaviors. The description of phenomena like morphological evolution, spinodal decomposition, and dislocation motion requires the description of material domains of several nm up to μm in length. The spatial scales involved are therefore often so large that atomistic approaches are not computationally feasible. This is true both for so-called *ab initio* methods, in which electrons are considered and the Schroedinger equation is solved (usually under some approximation, such as density functional theory, DFT), and for classical interatomic potentials, which parametrize the interaction energy between nuclei with some analytical expression. For this reason, we must resort to a continuum description of elastic and plastic deformations in materials. This can be done efficiently in the context of the classical field of linear elasticity. The main actor in linear elasticity theory is the *displacement field*, a vector field describing how material elements are displaced inside a continuum, deformed body. From that, two tensors can be derived, the strain field and the stress field, which describe the local deformation and body forces respectively. These two ingredients can be combined to describe the elastic energy stored in a deformed body.

B.1 THE STRAIN TENSOR

We start this quick revision of the theory of elasticity in a bottom-up approach. Consider a solid body made up by individual "bits", like atoms, molecules, etc. If the body is deformed by an external action, we can associate every fundamental bit with a *displacement* vector that points from the original, undeformed position to the new one. This sort-of-atomistic description may be converted to a continuum approach by "zooming out" and considering that every point in space is associated with a vector $\vec{u}$ obtained by averaging the displacement vectors of the objects inside a small volume (on the continuum scale) that contains many fundamental bits. Therefore, we can describe the deformation state of a solid through such *displacement vector field*.

In the linear theory of elasticity, the assumption that deformations are small is made. This is equivalent to a first-order approximation: during a deformation, the displacement of a volume element from the initial position $\vec{x}$ to the final position $\vec{x'}$ can be written as:

$$x_i' = x_i + u_i(\vec{x}) = x_i + \frac{du_i}{dx_j} x_j + o(x_j^2) \tag{B.1}$$

repeated indices imply summation over those indices, as is standard in Einstein notation. i and j index Cartesian directions. Since we are interested in

small deformations, we can suppress second and greater-order terms (hence the term *linear* in linear elasticity theory), yielding:

$$\Delta x = x_i' - x_i = \frac{\partial u_i}{\partial x_j} x_j \tag{B.2}$$

which in vector form reads

$$\vec{\Delta x} = \bar{J}\,\vec{x} \tag{B.3}$$

$\bar{J}$ is the Jacobian matrix of the transformation of the coordinate system from the undeformed state to the deformed one, and its components may be written as

$$J_{ij} = \frac{\partial u_i}{\partial x_j} \tag{B.4}$$

Its interpretation is the following: given the coordinate vector of a volume element before the deformation, the vector $\vec{\Delta x}$, obtained through $\bar{J}$ as in equation B.3, will be its displacement. Knowing the value of $\bar{J}$ at every point will therefore fully characterize the deformation of the solid in such a linear framework.

It is straightforward to derive some properties of the Jacobian matrix. First, being the gradient of a vector field, it transforms accordingly, thus behaving as a second-order tensor.

Another important property is the symmetry. $\bar{J}$ can be decomposed in the following way:

$$J_{ij} = \frac{\partial u_i}{\partial u_j} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) = \varepsilon_{ij} + \Omega_{ij} \tag{B.5}$$

This is a standard result in tensor analysis: any second-rank tensor may be simply expressed as the sum of an explicitly symmetric and an explicitly anti-symmetric tensor. Indeed,

$$\begin{cases} \varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \varepsilon_{ji} \\ \Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) = -\Omega_{ji} \end{cases} \tag{B.6}$$

The important point is the physical meaning of these two tensors. The second one Ω is called the *local rotations tensor* and represents rotations inside the material. Notice, however, that in the context of linear elasticity, we are interested in deformations. $\overline{\Omega}$ can therefore be neglected[1] and we can deal only with $\bar{\varepsilon}$. $\bar{\varepsilon}$ is one of the main ingredients in linear elasticity and is called *strain tensor*.

The fact that the strain tensor is symmetric allows us to make some simplifications. The most important one is that the strain state of a system can be specified by knowing only 6 independent components of the tensor, for example, the diagonal elements and the upper off-diagonal ones. This is

1 See e.g. Reference [48, 50] for theoretical justifications.

Figure 66: Simple expansion of a bar

commonly exploited in the so-called *Voigt notation*, in which a second-rank symmetric tensor may be formally written as a 6D vector:

$$\overline{\varepsilon} = \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{pmatrix} \rightarrow \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{yz} \\ \varepsilon_{xz} \\ \varepsilon_{xy} \end{pmatrix} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix} \tag{B.7}$$

where in the right equality Cartesian indices have been replaced by the conventional numeration.

The physical meaning of components of the strain tensor can be understood by looking at simple deformation conditions. Consider for example elongation in one direction, e.g. along the $\hat{x}$ direction, of a solid bar (the same logic applies for compression). Let us also fix the frame of reference so that the origin is at the beginning of the bar, as reported in Figure 66. After the deformation, the final point will be displaced by Δl from the point l where the bar initially ended. If the material is homogeneous and the deformation is evenly distributed along the bar, the displacement field will be:

$$\vec{u} = \begin{pmatrix} \frac{\Delta l}{l} x \\ 0 \\ 0 \end{pmatrix} \tag{B.8}$$

In this way the point initially at the origin ($x = 0$) will still be there, and the point at the end of the bar will move by $\frac{\Delta l}{l} l = \Delta l$, as expected, the middle point at $x = l/2$ will have "absorbed" half of the deformation and so on.

The strain tensor may now be obtained directly by the definition in equation B.6:

$$\overline{\varepsilon} = \begin{pmatrix} \frac{\Delta l}{l} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{B.9}$$

This straightforward example shows that the diagonal components of the strain tensor represent the relative compression/elongation of the solid. No-

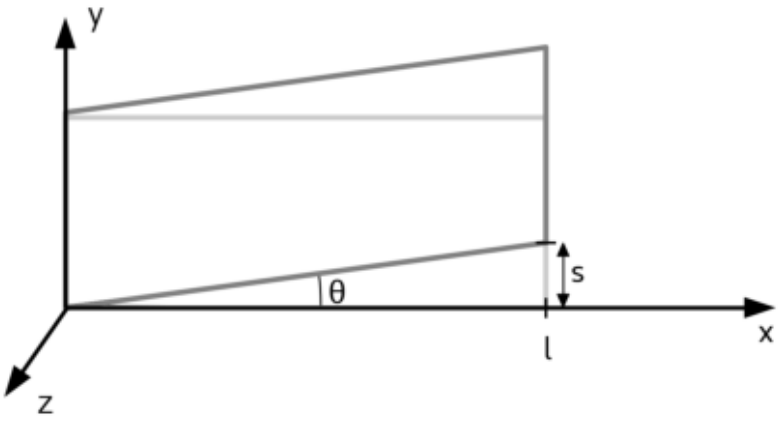

Figure 67: Simple shear deformation

tice that, by the convention used here, positive diagonal stress components mean that the body has been elongated, while negative diagonal stress components mean the body has been compressed.

To understand the meaning of off-diagonal strain components, let us consider a pure shear deformation. We position the origin of the frame of reference on the left plane and let the right plane slide in the positive y direction, keeping the inter-planar distance fixed (see figure 67). Indicating with s the distance traveled by the right plane, the displacement field is simply

$$\vec{u} = \begin{pmatrix} 0 \\ \frac{s}{l}x \\ 0 \end{pmatrix} \tag{B.10}$$

Using equation B.6, the strain field will be

$$\bar{\varepsilon} = \begin{pmatrix} 0 & \frac{s}{2l} & 0 \\ \frac{s}{2l} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{B.11}$$

Off-diagonal elements of the strain tensor represent therefore half the shear angle, since $\frac{s}{l} = \tan\theta \cong \theta$ (using small angle approximation, consistent with small deformation assumption).

B.2 THE STRESS TENSOR

We move now to the second fundamental object in the theory of elasticity, the *stress tensor*. When a body is deformed, its atoms and molecules will be displaced from their equilibrium positions, hence non-vanishing internal forces will be present. These forces are directed so to restore the undeformed configuration of the solid. By averaging the restoring forces in an infinitesimal volume element, we can obtain a force density $\vec{f}$ vector field. We can

obtain the total force in the i-th direction F_i acting on a portion of the solid of volume V as

$$F_i = \int_V f_i(\vec{x}) d^3\vec{x}$$

In principle, integration should be with respect to spatial variables in the deformed state. If the strain field is small (i.e. if the body is not severely deformed, which is the base assumption of linear elasticity), the difference in the two integrals vanishes.

Using Gauss' theorem to convert the volume integral into a surface integral on the boundary of V:

$$F_i = \int_V f_i(\vec{x}) d^3\vec{x} = \int_{\partial V} \vec{\tau_i}(\vec{x}) \cdot \hat{n} dS \tag{B.12}$$

where $\vec{\tau_i}$ is a vector field whose divergence is f_i ($\vec{\nabla} \cdot \vec{\tau_i} = f_i$) and $\hat{n}$ is the surface's normal vector.

Re-framing everything in vector form:

$$\vec{F} = \int_V \vec{f}(\vec{x}) d^3\vec{x} = \int_{\partial V} \overline{\sigma} \cdot \hat{n} dS \tag{B.13}$$

$\overline{\sigma}$ is a second-rank tensor whose rows are given by $\vec{\tau_i}$, called the *stress tensor*. Its meaning may be understood by inspecting the meaning of the right integrand in equation B.13: $\overline{\sigma}\hat{n}$ is the force per unit *area* acting on a surface whose normal is in the $\hat{n}$ direction. Suppose that $\hat{n} = \hat{x}$ (other components may be obtained by indices' permutation). Then

$$\overline{\sigma}\hat{x} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yx} \\ \sigma_{zx} \end{pmatrix} \tag{B.14}$$

which means that σ_{ij} is the force per unit area in the i-th direction acting on the surface normal to the j-th direction of a volume element. Diagonal elements are called hydrostatic terms, while off-diagonal terms are shear ones. The meaning of these names will be clear once the connection between the stress and the strain field is introduced. Notice that, by definition, a positive diagonal term means that the volume element is pulled *outward*, meaning that positive diagonal stress components are expansion ones (same as for strains).

Suppose that we are interested in a general elastic problem involving free surfaces. From the straightforward interpretation of the stress tensor, if no forces act on surfaces it must hold that:

$$\overline{\sigma}\hat{n} = \vec{0} \tag{B.15}$$

where $\hat{n}$ is the unit vector out of the free surface. This condition translates to a Neumann boundary condition when equations of elasticity are solved, both analytically and numerically.

B.2.1 Stress symmetries

As we did for strain, it's possible to find symmetries in the stress tensor. Suppose we want to calculate the total torque $\vec{T}$ acting on a volume V of the body. For simplicity, let's consider the x component (again, by permutation of indices it is possible to obtain results relating to other components). Then

$$T_x = \int_V y f_z - z f_y \, d^3\vec{x} = \int_V y\left(\frac{\partial \sigma_{zi}}{\partial x_i}\right) - z\left(\frac{\partial \sigma_{yi}}{\partial x_i}\right) \tag{B.16}$$

where summation on the partial derivatives is implied by Einstein's notation. We can now use the chain rule to obtain

$$T_x = \int_V \frac{\partial}{\partial x_i}(y\sigma_{zi} - z\sigma_{yi}) d^3\vec{x} - \int_V \sigma_{zi}\frac{\partial y}{\partial x_i} - \sigma_{yi}\frac{\partial z}{\partial x_i} d^3\vec{x} \tag{B.17}$$

Applying the divergence theorem to transform the first integral in a surface one we obtain:

$$T_x = \int_{\partial V} (y\sigma_{zi} - z\sigma_{yi}) n_i dS - \int_V \sigma_{zy} - \sigma_{yz} d^3\vec{x} \tag{B.18}$$

Here n_i is the i-th component of the normal vector of the volume's boundary.

Let's inspect the physical meaning of the terms in equation B.18. The first one is a surface term, meaning that it represents, by Newton's third law, the (equal and opposite) reaction to the external torque. The second one is the torque contribution due to the internal stresses of the body. However, Newton's third law of motion implies that this contribution must be zero[2]. This means that

$$\sigma_{ij} = \sigma_{ji} \tag{B.19}$$

hence the stress tensor is *symmetric* too and can be described by a 6D "vector" using Voigt notation.

B.2.2 Hooke law

Up to this point, we have defined stress and strain tensors which describe how a body deforms and which forces are acting on volume elements. What is missing is some function relating one to the other. This is called the *constitutive relation*. Since strain should be small, the most common approximation (which is also very convenient from an analytical point of view) is that of linearity:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{B.20}$$

Since the *elastic constant tensor* (or *elastic tensor*) $\overline{\overline{C}}$ is relating to rank-two tensors, it must be a rank-four tensor itself. The relationship between the

2 This is due to the assumption that the internal stresses are assumed to be fully characterized by *contact forces* between volume elements of the solids. See [48], §2.

strain and stress can also be expressed by the inverse elastic constant tensor, called the *compliance tensor* $\overline{\overline{S}}$.

As an effect of the symmetric nature of the stress and strain tensors, we can already say that

$$\begin{cases} C_{ijkl} = C_{jikl} \\ C_{ijkl} = C_{ijlk} \end{cases} \tag{B.21}$$

This means that the elastic tensor has only 36 independent components and using Voigt notation, it may be represented as a 6×6 matrix relating the 6D strain and stress "vector". The number of independent constants, however, may be further reduced once we define the elastic energy of a deformed body.

B.2.3 Elastic equilibrium

Now that we have both defined the meaning of the stress field and drawn a connection with the strain field (hence with the displacement $\vec{u}$), it is possible to derive the differential equations for the mechanical equilibrium in a deformed body. By definition (Equation B.12), the divergence of the stress field gives volume forces present in the solid. Writing Newton's equations of motion is therefore straightforward:

$$\rho \partial_t^2 \vec{u} = \vec{\nabla} \cdot \overline{\sigma} + \vec{f} \tag{B.22}$$

where ρ is the material density, ∂_t represents time derivative, $\vec{\nabla}\cdot$ is the divergence operator acting on the stress tensor, and $\vec{f}$ is any additional body force (e.g. due to gravity).

Equation B.22 can be made explicit in $\vec{u}$ utilizing the elastic tensor and definition B.6. Switching to Einstein's notation:

$$\rho \partial_t^2 u_i = \frac{\partial}{\partial x_j} \sigma_{ij} = C_{ijkl} \frac{\partial}{\partial x_j} \varepsilon_{kl} = \frac{1}{2} C_{ijkl} \frac{\partial}{\partial x_j} (u_k + u_l) \tag{B.23}$$

Of course, the solution of Eq. B.23 requires boundary conditions. Along with standard Dirichlet boundaries on $\vec{u}$, relationship B.15 defines Neumann boundary conditions.

Stress/strain fields at mechanical equilibrium can be found by enforcing Newton's first law:

$$\begin{cases} \vec{\nabla} \cdot \overline{\sigma} = -\vec{f} & \\ \overline{\sigma} \cdot \hat{n} = -\vec{T} & \text{on } \partial\Omega_N \\ \vec{u} = \vec{u}_0 & \text{on } \partial\Omega_D \end{cases} \tag{B.24}$$

where $\vec{T}$ are external tractions, $\partial\Omega_N$ represent Neumann boundary points, $\vec{u}_0$ is the displacement field at Dirichlet boundary points $\partial\Omega_D$. As in the

whole thesis, surface tractions, additional body forces and displacement at boundaries will vanish, we will use:

$$\begin{cases} \vec{\nabla} \cdot \overline{\sigma} = \vec{0} & \\ \overline{\sigma} \cdot \hat{n} = \vec{0} & \text{on } \partial\Omega_N \\ \vec{u} = \vec{0} & \text{on } \partial\Omega_D \end{cases} \tag{B.25}$$

B.3 ELASTIC ENERGY

Once the constitutive relation is defined, it is possible to describe the elastic energy of a linear solid. Consider the work density dw done by unit volume during an infinitesimal deformation process:

$$\delta w = f_i \delta u_i = \frac{\partial \sigma_{ij}}{\partial x_j} \delta u_i = \frac{\partial}{\partial x_j}(\sigma_{ij} \delta u_i) - \sigma_{ij} \delta \frac{\partial u_i}{\partial x_j} \tag{B.26}$$

The contribution of the first term may be evaluated using the divergence theorem and transformed into a surface contribution:

$$\int_V \frac{\partial}{\partial x_j}(\sigma_{ij} \delta u_i) d^3\vec{x} = \int_{\partial V} \sigma_{ij} \delta u_i dS \tag{B.27}$$

If we consider an unbounded solid that has no deformation at infinity, the integral vanishes. On the other hand, if the solid is finite and no external force is acting on it, surface stresses will be zero, and the term contribution may be neglected. These are the only situations that will be considered in this Thesis.

Using the symmetries of the stress tensor, the second term in equation B.26 may instead be rewritten as

$$\sigma_{ij} \delta \frac{\partial u_i}{\partial x_j} = \frac{1}{2} \sigma_{ij} \delta \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) = \sigma_{ij} \delta \varepsilon_{ij} \tag{B.28}$$

By equations B.20 and B.28 the variation in the free energy F of the body during a deformation will be

$$dF = \sigma_{ij} \delta \varepsilon_{ij} = C_{ijkl} \varepsilon_{kl} \delta \varepsilon_{ij} \tag{B.29}$$

which leads, through integration, to the elastic energy $U(\overline{\varepsilon})$

$$U(\overline{\varepsilon}) = \frac{1}{2} C_{ijkl} \varepsilon_{kl} \varepsilon_{ij} \tag{B.30}$$

which in tensor form reads

$$U(\overline{\varepsilon}) = \frac{1}{2} \overline{\overline{C}} \overline{\varepsilon} : \overline{\varepsilon} = \frac{1}{2} \overline{\sigma} : \overline{\varepsilon} \tag{B.31}$$

where the : operator indicates the sum of element-wise products. Notice that if term B.27 is retained (e.g. no vanishing tractions at the surface), a surface term is missing in the expression for the total energy, which amounts to the work done by the external traction in deforming the body surface.

Result B.31 is reminiscent of the simple harmonic oscillator energy $\frac{1}{2}kx^2$ and can be used to calculate the dislocation energy of the system (see Chapter 2) or the elastic contribution to chemical potentials (see Chap. 3).

The so-called *great symmetry* in the elastic constants tensor can be easily obtained considering the elastic energy density. It is straightforward to see that

$$C_{ijkl} = \frac{\partial^2 F}{\partial \varepsilon_{ij} \varepsilon_{kl}} \tag{B.32}$$

however, as the energy function is continuous, the order of differentiation doesn't matter for the result:

$$C_{ijkl} = C_{klij} \tag{B.33}$$

The number of independent elastic constants is therefore reduced to 21 and $\overline{\overline{C}}$ will be represented by a symmetric matrix in Voigt notation.

B.3.1 Interaction energy between strain sources

In general, the elastic energy stored in the body is easily calculated directly using equation B.31. In specific situations, however, the linearity underlying elastic theory can be exploited to decompose the total energy in different contributions. Suppose that in an elastic solid there are multiple *strain sources*. The term "source" is vague on purpose: strain sources can be inclusions, surface tractions, dislocations, body forces, etc. The only requirement is that strain sources have a strain (or stress) field associated with them.

Due to linearity, if N strain sources are present in the solid, the total strain field will be

$$\overline{\varepsilon}_{tot} = \sum_{i=1}^{N} \overline{\varepsilon}_i \tag{B.34}$$

where ε_i is the strain field associated with source i. Since in linear elasticity the principle of superposition applies, we may write the total elastic energy stored in the body:

$$U(\overline{\varepsilon}) = \frac{1}{2}\overline{\sigma}_{tot} : \overline{\varepsilon}_{tot} = \frac{1}{2}\sum_{i=1}^{N} \overline{\sigma}_i : \sum_{j=1}^{N} \overline{\varepsilon}_j = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \overline{\sigma}_i : \overline{\varepsilon}_j \tag{B.35}$$

Diagonal terms $\overline{\sigma}_i : \overline{\varepsilon}_i$ in expression B.31 correspond to the energy density that source i would have if it is the only present in the body and are usually referred as *self-energy* terms, in analogy with electrostatics. This is not the whole story, however: non-diagonal terms $\overline{\sigma}_i : \overline{\varepsilon}_j$ are also present, which amount to *interaction energy* terms. Notice that the decomposition is not unique, as multiple strain sources can be merged into a single, comprehensive source. It is therefore possible to adapt this procedure to the task at hand. Result B.35 will come in handy in Chapter 2.

B.4 NORMAL SOLIDS AND CRYSTAL SYMMETRIES

At no point in our previous discussion we considered possible symmetries in the molecular/atomic ordering of a solid. It is possible to prove that for cubic lattices there are at most 3 independent elastic constants, while for hexagonal lattices there are 5. The smaller number of independent components that can be considered is for *homogeneous and isotropic* materials, which are characterized by only 2 elastic constants. In the theory of linear elasticity, a solid with these two properties is called a *normal solid*.

While mono-crystalline materials are not isotropic, the normal solid approximation is mathematically convenient and is often enough to capture most of the relevant physics in simulations [171]. In this Thesis, we will assume that materials are isotropic.

More rigorously, a normal solid is a system that has the properties of linearity, isotropy and homogeneity, hence the elastic constants are the same at every point in the material and there is full rotational symmetry. Depending on the problem at hand, there are different ways to choose the two independent elastic constants. One of the most common choices is using Lamè constants λ (first Lamè constant) and μ (shear modulus). In Voigt notation:

$$\begin{pmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\\ \sigma_4\\ \sigma_5\\ \sigma_6\end{pmatrix} = \begin{pmatrix}2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0\\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0\\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0\\ 0 & 0 & 0 & 2\mu & 0 & 0\\ 0 & 0 & 0 & 0 & 2\mu & 0\\ 0 & 0 & 0 & 0 & 0 & 2\mu\end{pmatrix}\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \varepsilon_4\\ \varepsilon_5\\ \varepsilon_6\end{pmatrix} \tag{B.36}$$

Since the elastic energy is the quadratic form in the strains and the elastic constants, it can be proved directly that

$$\begin{aligned}U(\bar{\varepsilon}) =& \frac{1}{2}(\lambda+2\mu)(\varepsilon_{xx}+\varepsilon_{yy}+\varepsilon_{zz})^2 + \\ & 2\mu(\varepsilon_{xy}^2+\varepsilon_{xz}^2+\varepsilon_{yz}^2-\varepsilon_{xx}\varepsilon_{yy}-\varepsilon_{xx}\varepsilon_{zz}-\varepsilon_{yy}\varepsilon_{zz})\end{aligned} \tag{B.37}$$

Another common choice is the Poisson ratio ν and the Young modulus E. The first one is defined as (minus) the ratio between the elongation perpendicular to uni-axial stress and the compression along the stress (e.g. $\frac{\varepsilon_{yy}}{\varepsilon_{xx}}$, supposing σ_{xx} is the only stress). The second is given by the ratio between the uni-axial stress and the corresponding deformation (e.g. $\frac{\sigma_{xx}}{\varepsilon_{xx}}$). The relationship between Young-Poisson and Lamè variables is the following:

$$\begin{cases} E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu} \\ \nu = \frac{\lambda}{2(\lambda+\mu)} \end{cases} \tag{B.38}$$

Naturally, other sets of constants or even mixed descriptions are possible.

Using E and ν, the elastic energy can be written compactly in terms of the stress:

$$U(\overline{\sigma}) = \frac{1}{2E}(\sigma_{xx}+\sigma_{yy}+\sigma_{zz})^2 + \frac{1+\nu}{E}(\sigma_{xy}^2+\sigma_{xz}^2+\sigma_{yz}^2-\sigma_{xx}\sigma_{yy}-\sigma_{xx}\sigma_{zz}-\sigma_{yy}\sigma_{zz}) \tag{B.39}$$

B.5 THE EIGENSTRAIN FORMALISM

We close this Appendix with a discussion on the eigenstrain formalism. The main idea traces back to Eshelby [172] and, in particular, provides a flexible and general framework to obtain stress/strain fields associated with defects in a solid. Most importantly, the eigenstrain formalism allows us to tackle the problem of strained films of generic shapes and dislocations when difficult boundary conditions are present. We will not delve too deep into the eigenstrain theory, but we outline here the key concepts, as they will be used multiple times in the Thesis.

The key ingredient is a simple yet powerful decomposition of strains in a solid. Strains are divided into two different classes: *elastic strains*, which will be indicated with $\overline{e}$ following [51], and *non-elastic strains* or *eigenstrains*, which will be indicated by $\overline{\varepsilon}^*$. Eigenstrains are referred to as "stress-free strains". In other words, are strains that originate in the solid without an associated stress source. Importantly, $\overline{\varepsilon}^*$ can be used to describe plastic deformations inside a body. Defects (e.g. dislocations in Chap. 2), lattice mismatches in epitaxial systems (see Chap. 2 and Chap. 3), material deformations associated with phase transitions and thermal strains are some examples of eigenstrains.

ε^* need not to be homogeneous. When their support is compact, i.e. they are non-zero only in some closed and bounded regions of the solid, that region is referred to as an inclusion. Additionally, eigenstrains can be coupled with other fields present in the material. For example, the thermal strain will be a function of temperature, strains related to the mixing of chemical species will be coupled with a concentration field, and so on.

As eigenstrains do not carry an associated stress field, the constitutive relation for a solid will be modified accordingly:

$$\overline{\sigma} = \overline{\overline{C}}\overline{e} = \overline{\overline{C}}(\overline{\varepsilon}-\overline{\varepsilon}^*) \tag{B.40}$$

where $\overline{\varepsilon}$ is now the *total strain* in the material, which accounts for both elastic and non-elastic contribution. Still, definition B.6 applies to the total strain. This is because the displacement field describes atomic motion in the body irrespective of the motion origin. Notice that if the total strain is equal to the eigenstrain, the stress field vanishes. This serves as an additional interpretation of $\overline{\varepsilon}^*$: it is the strain field that, if applied to the body, would eliminate all residual stress present, e.g., due to the presence of thermal effects or defects.

The elastic energy stored in the body will also be modified accordingly:

$$U(\bar{\varepsilon}) = \frac{1}{2}\bar{\sigma} : \bar{e} = \frac{1}{2}\bar{\sigma} : (\bar{\varepsilon} - \bar{\varepsilon}^*) = \frac{1}{2}\bar{\bar{C}}(\bar{\varepsilon} - \bar{\varepsilon}^*) : (\bar{\varepsilon} - \bar{\varepsilon}^*) \tag{B.41}$$

Once eigenstrains are prescribed to the solid, material response in terms of the stress field can be calculated considering the modified elastic equilibrium equations:

$$\begin{cases} \vec{\nabla} \cdot \bar{\bar{C}}(\bar{\varepsilon} - \bar{\varepsilon}^*) = \vec{0} & \\ \bar{\sigma} \cdot \hat{n} = \vec{0} & \text{on } \partial\Omega_N \\ \vec{u} = \vec{0} & \text{on } \partial\Omega_D \end{cases} \tag{B.42}$$

where we have assumed no traction and no displacement conditions at boundaries. Reordering the first equation and introducing the vector field $\vec{\beta} = \vec{\nabla} \cdot \bar{\bar{C}}\bar{\varepsilon}^*$:

$$\vec{\nabla} \cdot \bar{\bar{C}}\bar{\varepsilon} = \vec{\beta} \tag{B.43}$$

If we compare Equation B.43 with the first expression in the elastic equilibrium equations B.24, we can recognize that $\vec{\beta}$, which in turn originates from the eigenstrain, is equivalent to a body force field. If $\bar{\varepsilon}^*$ are known, then, it is possible to solve equations B.42 using standard methods in linear elasticity, such as Green function spectral methods or Finite Element Method solvers and obtain the total displacement field. From there, a simple application of Equation B.42 yields the elastic energy stored in the body. Notice that, if multiple sources of eigenstrain are present (for example, multiple defects are present in the solid), then decomposition B.35 can be used.

C DISLOCATION ARRAY STRESS FIELDS

In this appendix, non-singular periodic stress fields for dislocation arrays with Burgers vector $\vec{b} = [b_x, b_y, b_z]$ are reported. L is the periodicity of the array and the dislocation line is along $\hat{z}$. Results have been taken from Refs. [65, 62, 67].

We define

$$Y_t = \sqrt{\frac{y^2 + t^2}{L^2}}$$

where t is a normalisation parameter representing a typical core radius (order of Å) introduced by Cai. In all simulations in this thesis work we used $t = 0.1218$ nm.

$$\begin{aligned}
\sigma_{xx} =& \frac{-\mu b_x}{2L}\Big[\frac{3y\pi}{LY_t}\frac{\sinh 2\pi Y_t}{\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}} \\
&- \frac{2y^3}{4L^3Y_t^3}\frac{-4\pi Y_t + 2\cos 2\pi\frac{x}{L}[2\pi Y_t \cosh 2\pi Y_t - \sinh 2\pi Y_t] + \sinh 4\pi Y_t}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2}\Big] \\
&+ \frac{\mu b_y}{2(1-\nu)L}\sin 2\pi\frac{x}{L}\Big[\frac{\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L} + \frac{2\pi y^2}{L^2 Y_t}\sinh 2\pi Y_t}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2}\Big] \\
\sigma_{yy} =& -\frac{\mu b_x \pi y}{(1-\nu)L^2}\frac{\cosh 2\pi Y_t \cos 2\pi\frac{x}{L} - 1}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2} \\
&+ \frac{\mu b_y}{2(1-\nu)L}\sin 2\pi\frac{x}{L}\Big[\frac{\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L} + 2\pi Y_t \sinh 2\pi Y_t}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2}\Big] \\
\sigma_{xy} =& \frac{\mu b_x}{2L^3(1-\nu)}\sin 2\pi\frac{x}{L}\Big[\frac{2\pi t^2 \sinh 2\pi Y_t}{Y_t[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2} \\
&+ \frac{L^2[\cos 2\pi Y_t - \cos 2\pi\frac{x}{L} - 2\pi Y_t \sinh 2\pi Y_t]}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2}\Big] \\
&- \frac{\mu b_y \pi y}{2(1-\nu)L^2}\frac{\cosh 2\pi Y_t \cos 2\pi\frac{x}{L} - 1}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2} \\
\sigma_{xz} =& -\frac{\mu b_z}{2L}\Big[\frac{y}{LY_t}\frac{\sinh 2\pi Y_t}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]} \\
&+ \frac{yt^2}{4L^3Y_t^3}\frac{-4\pi Y_t + 2\cos 2\pi\frac{x}{L}[2\pi Y_t \cosh 2\pi Y_t - \sinh 2\pi Y_t] + \sinh 4\pi Y_t}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2}\Big] \\
\sigma_{yz} =& \frac{\mu b_z}{2L}\sin 2\pi\frac{x}{L}\Big[\frac{\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L} + \frac{\pi t^2}{L^2 Y_t}\sinh 2\pi Y_t}{[\cosh 2\pi Y_t - \cos 2\pi\frac{x}{L}]^2}\Big]
\end{aligned}$$

D ATTACHMENT-DETACHMENT SURFACE EVOLUTION

We describe here how equations for profiles evolving due to attachment and detachment of atoms from the solid surroundings (in a non-diffusion-limited regime) can be obtained, while in Section 3.1.2 we described surface diffusion-limited dynamics. We will closely follow the original treatment from Mullins [99].

Let us consider a solid, non-flat free surface in contact with a gas phase. From basic thermodynamics, the chemical potential (per particle) of an ideal gas may be written as:

$$\mu_{gas} = k_B T \log \frac{p}{p_0} \tag{D.1}$$

where k_B is Boltzmann constant, T is temperature, p is the system pressure and p_0 is the reference pressure value when the free surface is flat. If local equilibrium is imposed at all points at the solid surface, the chemical potentials in the two phases should be equal:

$$\mu_{gas} = \mu_{solid} \implies k_B T \log \frac{p}{p_0} = \kappa \gamma V_a \tag{D.2}$$

where V_a is the conversion factor from free energy per unit length to energy per unit particle. Notice that dimensional analysis implies that V_a is measured in volume per particle, hence may be regarded as the atomic volume.

If we suppose that the pressure induced by surface curvature is small, $p \approx p_0$ we may expand the logarithm to first order. By doing that and rearranging the previous equation we obtain

$$\Delta p = \frac{\kappa \gamma V_a}{k_B T} \tag{D.3}$$

where $\Delta p = p - p_0$. By kinetic theory, the pressure due to an ideal gas on a wall is a linear function of the flux of particles impinging per unit area and unit time θ. The extra amount of particles impinging on the surface is therefore:

$$\Delta \theta = \frac{\Delta p}{\sqrt{2\pi m k_B T}} = \frac{\kappa \gamma V_a}{\sqrt{2\pi m}(k_B T)^{\frac{3}{2}}} \tag{D.4}$$

where m is the gas particles mass and $\Delta\theta$ is the excess number of emitted particles from the surface with respect to the equilibrium, reference condition of the flat free surface. The dynamics of the system may be understood considering the sign of this term: if the free surface is convex, then $\kappa > 0$, and particles detach from the surface, leading to a receding boundary; if, on the contrary, the free surface is concave, $\kappa < 0$, particles attach to the surface and the boundary front advances. Assuming that the normal velocity $v_{\hat{n}}$ for the moving surface is linearly dependent on $\Delta\vartheta$. Merging all quanti-

ties which are independent of surface shape and surface energy in a single kinetic constant K:

$$v_{\hat{n}} = -K\kappa\gamma \tag{D.5}$$

At this point, we may consider the time dependence on the profile function: $h(x) \to h(x,t)$. The normal velocity and the time derivative of h are simply related by projection:

$$v_{\hat{n}} = \frac{1}{\sqrt{1+h'(x)^2}}\frac{dh}{dt} \tag{D.6}$$

which leads to the non-linear partial differential equation for $h(x,t)$:

$$\frac{\partial h}{\partial t} = -K\sqrt{1+h'(x)^2}\kappa\gamma = -K\gamma\frac{h''}{|1+h'^2|}$$

The non-linearity may be lifted if a small slope limit $h' \to 0$ is considered. In that case, the equation reduces to a heat equation, describing the profile smoothing:

$$\frac{\partial h}{\partial t} = K\gamma h'' \tag{D.7}$$

Notice that in the derivation, atomic diffusion on the free surface and in the gas phase is not considered. In other words, it is assumed that as soon as one atom attaches/detaches from the solid surface, the (local) equilibrium condition in the gas phase is instantaneously restored.

E GAN TRAINING PSEUDOCODE

For completeness, we report a pseudocode defining training a GAN with noise injection using the procedure described by Arjovsky and Bottou [155]. $\mathbb{1}_{25}$ indicates the identity in 25D, which is used as covariance matrix in the extraction of z.

```
tot_epochs ← set number of epochs ;
σ ← set additive noise standard deviation ;
G ← initialize Generator ;
D ← initialize Discriminator ;
for epoch in tot_epochs do
    for (x_t, x_{t+Δt}) in dataset do
        (ε_1, ε_2) ← samples from N(0, σ) ;
        x_t ← x_t + ε_1 ;
        x_{t+Δt} ← x_{t+Δt} + ε_2 ;
        for 20 iterations do
            z ← sample from N(0, 1_25) ;
            ε_3 ← sample from N(0, σ);
            x̃_{t+Δt} ← G(x_t, z) ;
            x̃_{t+Δt} ← x̃_{t+Δt} + ε_3 ;
            s̃ ← D(x̃_{t+Δt}, x_t) ;
            s ← D(x_{t+Δt,x_t}) ;
            L_D ← −1/2 [log(s) + log(1 − s̃)] ;
            Adam update for θ_D ;
        end
        s̃ ← D(x̃_{t+Δt}, x_t) ;
        L_G ← −log(s̃) ;
        Adam update for θ_G;
    end
    Save G and D
end
```

F NOISE INJECTION IN GANS AT RUNNING TIME

As we have discussed in the main text, the noise injection procedure for both Generator and Discriminator is critical to yield converging GANs. It is still not clear however what is the noise role at iterative generation time. The solution is that new ε noise should be added to the *inputs* of G every time a new state of the system is sampled:

$$\tilde{x}_{t+\Delta t} = G(\tilde{x}_t + \varepsilon, z) \tag{F.1}$$

The reason why noise should be added to G input is based on consistency: in training, all inputs of G are convolved with the Gaussian distribution, therefore it has been trained to specifically deal with this kind of data.

A slightly more formal argument can be made considering the Generator architecture. Both z and ε are extracted from Gaussian distributions, albeit the first in a 25D space and the second one in 1D. Let us now consider the first hidden layer of G which involves an affine transformation of the input vector:

$$\overline{W} \begin{bmatrix} x_t + \varepsilon \\ z \end{bmatrix} \tag{F.2}$$

where $\overline{W}$ represent an affine transformation and $[x_t + \varepsilon, z]^T$ is obtained as a simple concatenation of the previous state of the system (with the additional Gaussian noise) and the latent vector z. However, this operation can be simply rephrased as

$$\overline{W} \begin{bmatrix} x_t + \varepsilon \\ z \end{bmatrix} = \overline{W} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & \mathbb{1} \end{bmatrix} \begin{bmatrix} x_t \\ \varepsilon \\ z \end{bmatrix} = \overline{W'} \begin{bmatrix} x_t \\ \tilde{z} \end{bmatrix} \tag{F.3}$$

where $\mathbb{1}$ is the identity matrix for the latent space. In the second equality, we absorbed affine transformations into a single map $\overline{W'}$ and $[\varepsilon, z]$ into a single vector $\tilde{z}$. From this reformulation, it is clear that the addition of random noise ε is in some sense equivalent to considering a Generator with a specific first layer structure. On the other hand, not adding Gaussian noise to the input, i.e. setting $\varepsilon = 0$, should have a similar impact to making one component of the latent space deterministic. Indeed, in our testing, we verified that deviations of kinetic and stationary properties obtained by not adding Gaussian noise at sequence generation time and zeroing out one of the components of z are comparable and that the injection of ε in G inputs is always beneficial.

On the other hand, the addition of random noise to G output is required during training as a sort of regularization procedure *for the Discriminator* [173]. It is therefore not necessary to add an independent random variable ε to the generated $\tilde{x}_{t+\Delta t}$. Even though the particle position has always been

"smoothed out" in training, the unperturbed output of G is sharply peaked near discrete positions, as can be observed in Ref. [148]. This is to be expected, as the original GAN theory [149] now applies to the data and the Generator probability distributions *convolved* with the random noise Gaussian [155]. At convergence, we should therefore have that

$$p_G * \mathcal{N}(0, \sigma) \approx p_{data} * \mathcal{N}(0, \sigma) \tag{F.4}$$

$*$ being the convolution product. If σ is *small enough*, this implies $p_G \approx p_{data}$.

BIBLIOGRAPHY

[1] L. Zdeborová, "New tool in the box," *Nature Physics*, vol. 13, no. 5, pp. 420–421, 2017.

[2] L. Himanen, A. Geurts, A. S. Foster, and P. Rinke, "Data-Driven Materials Science: Status, Challenges, and Perspectives," *Advanced Science*, vol. 6, p. 1900808, nov 2019.

[3] Y. Zhang and X. Xu, "Machine learning specific heat capacities of nanofluids containing cuo and al203," *AIChE Journal*, vol. 67, no. 9, p. e17289, 2021.

[4] Y. Zhang and X. Xu, "Predictions of adsorption energies of methane-related species on Cu-based alloys through machine learning," *Machine Learning with Applications*, vol. 3, p. 100010, 2021.

[5] J. Yeom, T. Stan, S. Hong, and P. W. Voorhees, "Segmentation of experimental datasets via convolutional neural networks trained on phase field simulations," *Acta Materialia*, vol. 214, p. 116990, 2021.

[6] V. Stanev, K. Choudhary, A. G. Kusne, J. Paglione, and I. Takeuchi, "Artificial intelligence for search and discovery of quantum materials," *Communications Materials*, vol. 2, no. 1, pp. 1–11, 2021.

[7] K. Kaufmann and K. S. Vecchio, "Searching for high entropy alloys: A machine learning approach," *Acta Materialia*, vol. 198, pp. 178–222, 2020.

[8] K. Frydrych, K. Karimi, M. Pecelerowicz, R. Alvarez, F. J. Dominguez-Gutiérrez, F. Rovaris, and S. Papanikolaou, "Materials informatics for mechanical deformation: A review of applications and challenges," *Materials*, vol. 14, no. 19, 2021.

[9] E. H. Lee, W. Jiang, H. Alsalman, T. Low, and V. Cherkassky, "Methodological framework for materials discovery using machine learning," *Physical Review Materials*, vol. 6, p. 043802, apr 2022.

[10] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[11] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[12] G. H. Teichert and K. Garikipati, "Machine learning materials physics: Surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics," *Computer*

Methods in Applied Mechanics and Engineering, vol. 344, pp. 666–693, 2019.

[13] K. Yang, Y. Cao, Y. Zhang, S. Fan, M. Tang, D. Aberg, B. Sadigh, and F. Zhou, "Self-supervised learning and prediction of microstructure evolution with convolutional recurrent neural networks," *Patterns*, vol. 2, no. 5, p. 100243, 2021.

[14] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, pp. 422–440, June 2021.

[15] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Physical Review Letters*, vol. 98, no. 14, pp. 1–4, 2007.

[16] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," *The Journal of Chemical Physics*, vol. 134, no. 7, p. 074106, 2011.

[17] A. P. Bartõk and G. Csányi, "Gaussian approximation potentials: A brief tutorial introduction," *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1051–1057, 2015.

[18] G. Csányi, J. R. Kermode, S. De, N. Bernstein, M. Ceriotti, A. P. Bartók, and C. Poelking, "Machine learning unifies the modeling of materials and molecules," *Science Advances*, vol. 3, no. 12, p. e1701816, 2017.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[20] R. B. Ash, *Basic probability theory*. John Wiley & Sons, Inc., 1970.

[21] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[23] C. C. Aggarwal *et al.*, "Neural networks and deep learning," *Springer*, vol. 10, pp. 978–3, 2018.

[24] P. Mehta, M. Bukov, C. H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to Machine Learning for physicists," *Physics Reports*, vol. 810, pp. 1–124, 2019.

[25] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.

[26] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, no. 4, p. 045002, 2019.

[27] A. P. Bartók, M. P. Payne, R. Kondor, and G. Csányi, "Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons," *Physical Review Letters*, vol. 104, p. 136403, 2010.

[28] J. Behler, "Representing potential energy surfaces by high-dimensional neural network potentials," *Journal of Physics Condensed Matter*, vol. 26, no. 18, 2014.

[29] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi, "Gaussian Process Regression for Materials and Molecules," *Chemical Reviews*, vol. 121, no. 16, pp. 10073–10141, 2021.

[30] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[31] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[32] F. Boccardo and O. Pierre-Louis, "Controlling the shape of small clusters with and without macroscopic fields," *Phys. Rev. Lett.*, vol. 128, p. 256102, Jun 2022.

[33] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[34] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[36] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[37] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, pp. 462–466, 1952.

[38] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[39] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *International conference on machine learning*, pp. 1225–1234, PMLR, 2016.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[43] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[44] T. Tieleman and G. Hinton, "Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning," *COURSERA Neural Networks Mach. Learn*, vol. 17, 2012.

[45] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[46] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[47] L. Landau and E. Lifshitz, *Mechanics*. Elsevier, 1982.

[48] L. D. Landau and E. M. Lifsits, *Theory of Elasticity*. Elsevier, 1986.

[49] J. Hirth and J. Lothe, *Theory of Dislocations*. Krieger Publishing Company, 1982.

[50] L. Colombo and S. Giordano, eds., *Introduzione alla Teoria della elasticità*. UNITEXT, Milano: Springer Milan, 2007.

[51] T. Mura, *Micromechanics of defects in solids*. Springer Science & Business Media, 2013.

[52] F. Nabarro, *Theory of Crystal Dislocations*. International series of monographs on physics, Clarendon P., 1967.

[53] R. W. Lardner, *Mathematical theory of dislocations and fracture*. University of Toronto Press, 1974.

[54] V. Bulatov and W. Cai, *Computer simulations of dislocations*. Oxford New York: Oxford University Press, 2006.

[55] D. Lanzoni, F. Rovaris, and F. Montalenti, "Machine learning potential for interacting dislocations in the presence of free surfaces," *Scientific Reports*, vol. 12, no. 1, p. 3760, 2022.

[56] D. Hull and D. J. Bacon, *Introduction to dislocations*. Butterworth-Heinemann, 2001.

[57] R. Hull, "Equilibrium theory of misfit dislocations networks," in *Properties of Silicon Germanium and SiGe: Carbon* (E. Kasper and K. Lyutovich, eds.), pp. 21–41, Exeter (UK): Inspec, 2000.

[58] E. A. Fitzgerald, Y. H. Xie, M. L. Green, D. Brasen, A. R. Kortan, J. Michel, Y. J. Mii, and B. E. Weir, "Totally relaxed GexSi1-x layers with low threading dislocation densities grown on Si substrates," *Applied Physics Letters*, vol. 59, pp. 811–813, aug 1991.

[59] F. Montalenti, F. Rovaris, R. Bergamaschini, L. Miglio, M. Salvalaglio, G. Isella, F. Isa, and H. von Känel, "Dislocation-Free SiGe/Si Heterostructures," *Crystals*, vol. 8, p. 257, 2018.

[60] E. Kasper and K. Lyutovich, *Properties of silicon germanium and SiGe: carbon*. IET, 2000.

[61] E. Clouet, L. Ventelon, and F. Willaime, "Dislocation core energies and core fields from first principles," *Physical Review Letters*, vol. 102, no. 5, pp. 1–4, 2009.

[62] W. Cai, A. Arsenlis, C. R. Weinberger, and V. V. Bulatov, "A non-singular continuum theory of dislocations," *Journal of the Mechanics and Physics of Solids*, vol. 54, no. 3, pp. 561–587, 2006.

[63] D. J. Griffiths, *Introduction to Electrodynamics*. Cambridge University Press, 2017.

[64] A. K. Head, "Edge dislocations in inhomogeneous media," *Proceedings of the Physical Society. Section B*, vol. 66, no. 9, pp. 793–801, 1953.

[65] A. M. Andrews, R. LeSar, M. A. Kerner, J. S. Speck, A. E. Romanov, A. L. Kolesnikova, M. Bobeth, and W. Pompe, "Modeling crosshatch surface morphology in growing mismatched layers. Part II: Periodic boundary conditions and dislocation groups," *Journal of Applied Physics*, vol. 95, no. 11, pp. 6032–6047, 2004.

[66] F. Rovaris, M. H. Zoellner, P. Zaumseil, A. Marzegalli, L. Di Gaspare, M. De Seta, T. Schroeder, P. Storck, G. Schwalb, G. Capellini, and F. Montalenti, "Dynamics of crosshatch patterns in heteroepitaxy," *Phys. Rev. B*, vol. 100, p. 085307, Aug 2019.

[67] F. Rovaris, F. Montalenti, and M. Bernasconi, "Continuum modeling of heteroepitaxy at the mesoscale: tackling elastic and plastic relaxation," 2020.

[68] J. Tersoff, "Dislocations and strain relief in compositionally graded layers," *Applied Physics Letters*, vol. 62, no. 7, pp. 693–695, 1993.

[69] J. Tersoff, "Dislocations and strain relief in compositionally graded layers -Erratum," *Applied Physics Letters*, vol. 62, no. 7, pp. 693–695, 1993.

[70] D. Sidoti, S. Xhurxhi, T. Kujofsa, S. Cheruku, J. P. Correa, B. Bertoli, P. B. Rago, E. N. Suarez, F. C. Jain, and J. E. Ayers, "Initial misfit dislocations in a graded heteroepitaxial layer," *Journal of Applied Physics*, vol. 109, no. 2, 2011.

[71] A. Marzegalli, M. Brunetto, M. Salvalaglio, F. Montalenti, G. Nicotra, M. Scuderi, C. Spinella, M. De Seta, and G. Capellini, "Onset of plastic relaxation in the growth of Ge on Si(001) at low temperatures: Atomic-scale microscopy and dislocation modeling," *Physical Review B*, vol. 88, no. 16, pp. 1–6, 2013.

[72] D. Lanzoni, F. Rovaris, and F. Montalenti, "Computational analysis of low-energy dislocation configurations in graded layers," *Crystals*, vol. 10, no. 8, p. 661, 2020.

[73] R. Gatti, A. Marzegalli, V. A. Zinovyev, F. Montalenti, and L. Miglio, "Modeling the plastic relaxation onset in realistic SiGe islands on Si(001)," *Physical Review B - Condensed Matter and Materials Physics*, vol. 78, no. 18, pp. 1–12, 2008.

[74] F. Rovaris, R. Bergamaschini, and F. Montalenti, "Modeling the competition between elastic and plastic relaxation in semiconductor heteroepitaxy: From cyclic growth to flat films," *Physical Review B*, vol. 94, p. 205304, nov 2016.

[75] F. Rovaris, F. Isa, R. Gatti, A. Jung, G. Isella, F. Montalenti, and H. von Känel, "Three-dimensional SiGe/Si heterostructures: Switching the dislocation sign by substrate under-etching," *Physical Review Materials*, vol. 1, p. 073602, dec 2017.

[76] F. Rovaris, R. Bergamaschini, and F. Montalenti, "Modeling the competition between elastic and plastic relaxation in semiconductor heteroepitaxy: From cyclic growth to flat films," *Phys. Rev. B*, vol. 94, p. 205304, Nov 2016.

[77] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," *Physical Review Letters*, vol. 104, no. 13, pp. 1–4, 2010.

[78] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi, "Gaussian Process Regression for Materials and Molecules," *Chemical Reviews*, vol. 121, no. 16, pp. 10073–10141, 2021.

[79] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren, "Neural network models of potential energy surfaces," *The Journal of Chemical Physics*, vol. 103, pp. 4129–4137, 09 1995.

[80] J. Behler, "Four generations of high-dimensional neural network potentials," *Chemical Reviews*, vol. 121, no. 16, pp. 10037–10072, 2021. PMID: 33779150.

[81] Y. Shaidu, E. Küçükbenli, R. Lot, F. Pellegrini, E. Kaxiras, and S. de Gironcoli, "A systematic approach to generating accurate neural network potentials: the case of carbon," *npj Computational Materials*, vol. 7, no. 1, pp. 1–13, 2021.

[82] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids*. Oxford university press, 2017.

[83] A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," *Phys. Rev. B*, vol. 87, p. 184115, May 2013.

[84] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, "Sobolev training for neural networks," *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 4279–4288, 2017.

[85] M. Kissel and K. Diepold, "Sobolev Training with Approximated Derivatives for Black-Box Function Regression with Neural Networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11907 LNAI, pp. 399–414, 2020.

[86] A. Sakai, N. Taoka, O. Nakatsuka, S. Zaima, and Y. Yasuda, "Pure-edge dislocation network for strain-relaxed SiGe/Si(001) systems," *Applied Physics Letters*, vol. 86, no. 22, p. 221916, 2005.

[87] N. Taoka, A. Sakai, T. Egawa, O. Nakatsuka, S. Zaima, and Y. Yasuda, "Growth and characterization of strain-relaxed SiGe buffer layers on Si(001) substrates with pure-edge misfit dislocations," *Materials Science in Semiconductor Processing*, vol. 8, pp. 131–135, feb 2005.

[88] F. K. LeGoues, M. C. Reuter, J. Tersoff, M. Hammar, and R. M. Tromp, "Cyclic growth of strain-relaxed islands," *Phys. Rev. Lett.*, vol. 73, pp. 300–303, Jul 1994.

[89] A. Marzegalli, V. A. Zinovyev, F. Montalenti, A. Rastelli, M. Stoffel, T. Merdzhanova, O. G. Schmidt, and L. Miglio, "Critical Shape and Size for Dislocation Nucleation in Si1-xGex Islands on Si(001) A.," *Physical Review Letters*, vol. 99, no. 23, p. 235505, 2007.

[90] M. Salvalaglio and F. Montalenti, "Fine control of plastic and elastic relaxation in Ge/Si vertical heterostructures," *Journal of Applied Physics*, vol. 116, p. 104306, sep 2014.

[91] F. Isa, M. Salvalaglio, Y. A. R. Dasilva, A. Jung, G. Isella, R. Erni, P. Niedermann, P. Gröning, F. Montalenti, and H. von Känel, "From plastic to elastic stress relaxation in highly mismatched SiGe/Si heterostructures," *Acta Materialia*, vol. 114, pp. 97–105, 2016.

[92] F. Isa, M. Salvalaglio, Y. A. R. Dasilva, M. Meduňa, M. Barget, A. Jung, T. Kreiliger, G. Isella, R. Erni, F. Pezzoli, E. Bonera, P. Niedermann, P. Gröning, F. Montalenti, and H. von Känel, "Highly Mismatched, Dislocation-Free SiGe/Si Heterostructures," *Advanced Materials*, vol. 28, no. 5, pp. 884–888, 2016.

[93] F. Rovaris, F. Isa, R. Gatti, A. Jung, G. Isella, F. Montalenti, and H. von Känel, "Three-dimensional SiGe/Si heterostructures: Switching the dislocation sign by substrate under-etching," *Physical Review Materials*, vol. 1, p. 073602, dec 2017.

[94] G. Rossi and R. Ferrando, "Searching for low-energy structures of nanoparticles: A comparison of different methods and algorithms," *Journal of Physics Condensed Matter*, vol. 21, no. 8, 2009.

[95] N. Taoka, A. Sakai, T. Egawa, O. Nakatsuka, S. Zaima, and Y. Yasuda, "Growth and characterization of strain-relaxed SiGe buffer layers on Si(001) substrates with pure-edge misfit dislocations," *Materials Science in Semiconductor Processing*, vol. 8, pp. 131–135, feb 2005.

[96] B. J. Spencer and J. Tersoff, "Stresses and first-order dislocation energetics in equilibrium stranski-krastanow islands," *Phys. Rev. B*, vol. 63, p. 205424, May 2001.

[97] F. Boioli, V. A. Zinovyev, R. Gatti, A. Marzegalli, F. Montalenti, M. Stoffel, T. Merdzhanova, L. Wang, F. Pezzoli, A. Rastelli, O. G. Schmidt, and L. Miglio, "Self-Ordering of Misfit Dislocation Segments in Epitaxial SiGe Islands on Si(001)," *Journal of Applied Physics*, vol. 110, no. 4, p. 044310, 2011.

[98] W. P. Kuykendall and W. Cai, "Conditional convergence in two-dimensional dislocation dynamics," *Modelling and Simulation in Materials Science and Engineering*, vol. 21, no. 5, 2013.

[99] W. W. Mullins, "Theory of Thermal Grooving," *Journal of Applied Physics*, vol. 28, no. 3, p. 333, 1957.

[100] L. Martín-Encinar, D. Lanzoni, A. Fantasia, F. Rovaris, R. Bergamaschini, and F. Montalenti, "Deep learning of surface elastic chemical potential in strained films: from statics to dynamics," 2024. Materials Cloud Archive 2024.X.

[101] V. A. Shchukin and D. Bimberg, "Spontaneous ordering of nanostructures on crystal surfaces," *Reviews of Modern Physics*, vol. 71, no. 4, p. 1125, 1999.

[102] J. Evans, P. Thiel, and M. C. Bartelt, "Morphological evolution during epitaxial thin film growth: Formation of 2d islands and 3d mounds," *Surface science reports*, vol. 61, no. 1-2, pp. 1–128, 2006.

[103] B. Li, J. Lowengrub, A. Rätz, and A. Voigt, "Review article: Geometric evolution laws for thin crystalline films: modeling and numerics," *Commun. Comput. Phys.*, vol. 6, no. 3, pp. 433–482, 2009.

[104] K. D. Brommer, M. Needels, B. Larson, and J. Joannopoulos, "Ab initio theory of the si (111)-(7× 7) surface reconstruction: A challenge for massively parallel computation," *Physical review letters*, vol. 68, no. 9, p. 1355, 1992.

[105] D. Chiappe, E. Scalise, E. Cinquanta, C. Grazianetti, B. van den Broek, M. Fanciulli, M. Houssa, and A. Molle, "Two-dimensional si nanosheets with local hexagonal structure on a mos2 surface," *Advanced Materials*, vol. 26, no. 13, pp. 2096–2101, 2014.

[106] F. Baletto, C. Mottet, and R. Ferrando, "Molecular dynamics simulations of surface diffusion and growth on silver and gold clusters," *Surface Science*, vol. 446, no. 1-2, pp. 31–45, 2000.

[107] J. Sprague, F. Montalenti, B. Uberuaga, J. Kress, and A. Voter, "Simulation of growth of cu on ag (001) at experimental deposition rates," *Physical Review B*, vol. 66, no. 20, p. 205415, 2002.

[108] R. Bergamaschini, F. Montalenti, and L. Miglio, "Optimal growth conditions for selective ge islands positioning on pit-patterned si (001)," *Nanoscale research letters*, vol. 5, pp. 1873–1877, 2010.

[109] J.-N. Aqua, I. Berbezier, L. Favre, T. Frisch, and A. Ronda, "Growth and self-organization of sige nanostructures," *Physics Reports*, vol. 522, no. 2, pp. 59–189, 2013.

[110] H. Kielhöfer, *Calculus of Variations*, vol. 67 of *Texts in Applied Mathematics*. Cham: Springer International Publishing, 2018.

[111] M. Perez, "Gibbs–thomson effects in phase transformations," *Scripta materialia*, vol. 52, no. 8, pp. 709–712, 2005.

[112] L. Onsager, "Reciprocal relations in irreversible processes. i.," *Phys. Rev.*, vol. 37, pp. 405–426, Feb 1931.

[113] D. Srolovitz, "On the stability of surfaces of stressed solids," *Acta Metallurgica*, vol. 37, no. 2, pp. 621–625, 1989.

[114] R. J. Asaro and W. A. Tiller, "Interface morphology development during stress corrosion cracking: Part i. via surface diffusion," *Metallurgical Transactions*, vol. 3, no. 7, pp. 1789–1796, 1972.

[115] M. Grinfeld, "Instability of the separation boundary between a nonhydrostatically stressed elastic body and a melt," in *Soviet Physics Doklady*, vol. 31, p. 831, 1986.

[116] M. Grinfeld, "The stress driven instability in elastic crystals: Mathematical models and physical manifestations," *Journal of Nonlinear Science*, vol. 3, pp. 35–83, 1993.

[117] G.-H. Lu and F. Liu, "Towards quantitative understanding of formation and stability of ge hut islands on si (001)," *Physical review letters*, vol. 94, no. 17, p. 176103, 2005.

[118] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[119] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, pp. 2352–2449, sep 2017.

[120] P. J. Davis, *Circulant matrices*, vol. 2. Wiley New York, 1979.

[121] F. W. Byron and R. W. Fuller, *Mathematics of classical and quantum physics*. Courier Corporation, 2012.

[122] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[123] H. Gao, L. Sun, and J.-X. Wang, "Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain," *Journal of Computational Physics*, vol. 428, p. 110079, 2021.

[124] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[125] K. Perlin, "Image synthesizer.," *Computer Graphics (ACM)*, vol. 19, no. 3, p. 287 – 296, 1985.

[126] "Python implementation for perlin noise." https://pypi.org/project/perlin-noise/. Accessed: 23-11-2023.

[127] J.-N. Aqua, T. Frisch, and A. Verga, "Nonlinear evolution of a morphological instability in a strained epitaxial film," *Phys. Rev. B*, vol. 76, p. 165319, Oct 2007.

[128] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[129] F. Boccardo, F. Rovaris, A. Tripathi, F. Montalenti, and O. Pierre-Louis, "Stress-Induced Acceleration and Ordering in Solid-State Dewetting," *Physical Review Letters*, vol. 128, p. 026101, jan 2022.

[130] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[131] N. Provatas and K. Elder, *Phase-field methods in materials science and engineering*. John Wiley & Sons, 2011.

[132] D. Lanzoni, M. Albani, R. Bergamaschini, and F. Montalenti, "Morphological evolution via surface diffusion learned by convolutional, recurrent neural networks: Extrapolation and prediction uncertainty," *Phys. Rev. Mater.*, vol. 6, no. 10, p. 103801, 2022.

[133] J. W. Cahn and J. E. Hilliard, "Free energy of a nonuniform system. i. interfacial free energy," *The Journal of chemical physics*, vol. 28, no. 2, pp. 258–267, 1958.

[134] J. W. Cahn, "On spinodal decomposition," *Acta metallurgica*, vol. 9, no. 9, pp. 795–801, 1961.

[135] J. Zhu, L.-Q. Chen, and J. Shen, "Morphological evolution during phase separation and coarsening with strong inhomogeneous elasticity," *Modelling and Simulation in Materials Science and Engineering*, vol. 9, no. 6, p. 499, 2001.

[136] P. C. Hiemenz and T. P. Lodge, *Polymer chemistry*. CRC press, 2007.

[137] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[138] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014.

[139] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 802–810, 2015.

[140] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–11, 2016.

[141] I. Peivaste, N. H. Siboni, G. Alahyarizadeh, R. Ghaderi, B. Svendsen, D. Raabe, and J. R. Mianroodi, "Machine-learning-based surrogate modeling of microstructure evolution using phase-field," *Computational Materials Science*, vol. 214, p. 111750, 2022.

[142] S. Vey and A. Voigt, "AMDiS: adaptive multidimensional simulations," *Comput. Vis. Sci.*, vol. 10, pp. 57–67, feb 2007.

[143] T. Witkowski, S. Ling, S. Praetorius, and A. Voigt, "Software concepts and numerical algorithms for a scalable adaptive parallel finite element method," *Advances in Computational Mathematics*, vol. 41, pp. 1145–1177, dec 2015.

[144] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[145] R. Tibshirani, "A Comparison of Some Error Estimates for Neural Network Models," *Neural Computation*, vol. 8, pp. 152–163, jan 1996.

[146] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[147] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[148] D. Lanzoni, O. Pierre-Louis, and F. Montalenti, "Accurate generation of stochastic dynamics based on multi-model generative adversarial networks," *The Journal of Chemical Physics*, vol. 159, no. 14, 2023.

[149] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[150] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[151] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[152] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[153] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[154] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *Advances in neural information processing systems*, vol. 29, 2016.

[155] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017. _eprint: 1701.04862.

[156] J. Li, A. Madry, J. Peebles, and L. Schmidt, "On the limitations of first-order approximation in gan dynamics," in *International Conference on Machine Learning*, pp. 3005–3013, PMLR, 2018.

[157] W. Nie and A. B. Patel, "Towards a better understanding and regularization of gan training dynamics," in *Uncertainty in Artificial Intelligence*, pp. 281–291, PMLR, 2020.

[158] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," 2017. _eprint: 1701.00160.

[159] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[160] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

[161] S. Kim, J. Noh, G. H. Gu, A. Aspuru-Guzik, and Y. Jung, "Generative adversarial networks for crystal structure prediction," *ACS central science*, vol. 6, no. 8, pp. 1412–1420, 2020. Publisher: ACS Publications.

[162] Y. Dan, Y. Zhao, X. Li, S. Li, M. Hu, and J. Hu, "Generative adversarial networks (GAN) based efficient sampling of chemical composition space for inverse design of inorganic materials," *npj Computational Materials*, vol. 6, no. 1, p. 84, 2020. Publisher: Nature Publishing Group UK London.

[163] P. Stinis, C. Daskalakis, and P. J. Atzberger, "SDYN-GANs: Adversarial learning methods for multistep generative models for general order stochastic dynamics," 2023. _eprint: 2302.03663.

[164] K. Yeo, Z. Li, and W. Gifford, "Generative adversarial network for probabilistic forecast of random dynamical systems," *SIAM J. Sci. Comput.*, vol. 44, no. 4, pp. A2150–A2175, 2022. Place: USA Publisher: Society for Industrial and Applied Mathematics.

[165] N. G. Van Kampen, *Stochastic processes in physics and chemistry*, vol. 1. Elsevier, 1992.

[166] A. F. Voter, "Introduction to the kinetic monte carlo method," in *Radiation effects in solids*, pp. 1–23, Springer, 2007.

[167] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[168] L. Gagliardi and O. Pierre-Louis, "Controlling anisotropy in 2d microscopic models of growth," *Journal of Computational Physics*, vol. 452, p. 110936, 2022.

[169] B. Marguet, F. A. Reis, and O. Pierre-Louis, "Interface collisions with diffusive mass transport," *Physical Review E*, vol. 106, no. 1, p. 014802, 2022.

[170] C. Misbah, O. Pierre-Louis, and Y. Saito, "Crystal surfaces in and out of equilibrium: A modern view," *Reviews of Modern Physics*, vol. 82, no. 1, p. 981, 2010.

[171] E. Fitzgerald, "Dislocations in strained-layer epitaxy: theory, experiment, and applications," *Materials Science Reports*, vol. 7, pp. 87–142, nov 1991.

[172] J. D. Eshelby, "The determination of the elastic field of an ellipsoidal inclusion, and related problems," *Proceedings of the royal society of London. Series A. Mathematical and physical sciences*, vol. 241, no. 1226, pp. 376–396, 1957.

[173] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.