Tampereen yliopisto

Mika Sarvilahti

PREDICTABILITY OF ONE-DIMENSIONAL DISLOCATION SYSTEMS

Master's Thesis
Faculty of Engineering and Natural Sciences
Examiner 1: Lasse Laurson
Examiner 2: Audun Skaugen
April 2020

ABSTRACT

Mika Sarvilahti: Predictability of One-Dimensional Dislocation Systems
Master's Thesis
Tampere University
Degree Programme in Engineering and Natural Sciences, MSc
April 2020

Plastic deformation processes can be modelled with computationally exhaustive simulations, but utilizing new machine learning techniques can reduce the workload and simultaneously provide new insight on the processes by revealing previously unknown dependencies between physical properties. In this work, one-dimensional periodic models of edge dislocation pileups interacting with fixed pinning landscapes are studied. Using information about the pinning landscape and about the initial dislocation state, predictive models such as linear regression, simple neural networks and convolutional neural networks are used to test how predictable the stress-strain curves obtained by simulations are. Predictability of avalanches, the critical events within the stress-strain curves, is tested separately as well.

Introductions to some of the physical background, statistical methods and machine learning techniques are given. A geometrical interpretation is shown for the correlation coefficient, which is important in this work since it is used to measure how well predicted estimates match with the desired targets. Predictive models are described as consecutive sets of operations that are thoroughly explained.

Choices made for simulations and predictive model training processes are discussed in detail. Notably, the L^1-regularization technique is found useful and is utilized whenever possible. For non-convolutional predictive models, quantile representations of distributions derived from the pinning landscape and from the initial dislocation state are used as the input features. The basic structure of the convolutional neural network models is specially customized for the purposes of this work's periodic systems, providing the prediction results full expected independence from any spatial shift on the model input.

Statistical analysis is performed on the simulation results before continuing to the assessment of predictability. Training of convolutional network models turns out more problematic and much slower than the simple regression models, but in turn, convolutional models give the best predictability results. Simple regression models train and estimate much faster but can still provide nearly as good predictions of the stress-strain curves as the convolutional models.

A prediction problem for predicting avalanches alone is defined. Neural network models can vaguely predict the appearance of avalanches when their size and the stress at which they occur are specific. Predictability of avalanches is found worse than predictability of stresses of stress-strain curves, although it should be noted that the prediction problems are defined quite differently.

Keywords: plastic deformation, machine learning, computational material physics, dislocations

TIIVISTELMÄ

Mika Sarvilahti: Ennustettavuus yksiulotteisissa dislokaatiosysteemeissä
Diplomityö
Tampereen yliopisto
Teknis-luonnontieteellinen DI-tutkinto-ohjelma
Huhtikuu 2020

Plastisia deformaatioprosesseja voidaan mallintaa laskennallisesti raskailla simulaatioilla, mutta uusien koneoppimismenetelmien avulla on mahdollista ennustaa simulaatioiden tuloksia tehokkaasti. Samalla voi oppia ymmärtämään prosesseja paremmin, kun paljastuu aiemmin tuntemattomia yhteyksiä materiaalien sisältämien eri ominaisuuksien välillä. Tässä työssä tutkitaan paikallaan olevan pinnausmaiseman (epäpuhtaus-/kidevirhemaiseman) kanssa vuorovaikuttavia särmädislokaatiokasaumia kuvaavia yksiulotteisia, periodisia malleja. Simulaatioiden tuottamien jännitysvoima-venymä—käyrien ennustettavuutta, tiedettäessä pinnausmaisema ja dislokaatioiden alkutila, testataan käyttämällä erilaisia ennustusmalleja, kuten lineaarista regressiota, yksinkertaisia neuroverkkoja ja konvoluutioneuroverkkoja. Myös jännitysvoima-venymä—käyrien sisältämien vyöryjen ennustettavuutta testataan erikseen.

Työssä pohjustetaan tuloksia esittelemällä työn kannalta merkityksellistä materiaalifysiikan taustateoriaa, tilastomatematiikan analysointikeinoja ja koneoppimismalleja. Korrelaatiokerrointa käytetään tässä työssä mittarina sille, kuinka hyvin ennustukset sopivat yhteen todellisten tavoitekohteiden kanssa, joten korrelaatiokertoimelle on valittu esitettäväksi geometrinen, mahdollisimman ymmärrettävä tulkinta. Samoin koneoppimismallien toimintaperiaatteet on pyritty selittämään ymmärrettävästi käyttämällä merkintätapana operaatioiden sarjoja.

Simulaatioita ja koneoppimisprosesseja varten tehdyt valinnat käydään yksikohtaisesti läpi. Koneoppimisen yhteydessä käytettävä L^1-regularisointitapa osoittautuu hyödylliseksi, ja sitä käytetäänkin tässä työssä aina kun mahdollista. Muiden kuin konvoluutioneuroverkkojen tapauksessa ennustusmallit käyttävät ennustamiseen kvantiileja, jotka kuvaavat pinnausmaisemasta ja dislokaatioiden alkutilasta johdettuja jakaumia. Konvoluutioneuroverkkojen rakenne on suunniteltu sopivaksi tätä työtä varten, jossa käsitellään periodisia systeemejä, joiden ominaisuuksien tiedetään jo ennalta olevan riippumattomia siitä, mistä kohdasta systeemiä tarkastellaan.

Ennen kuin edetään varsinaisiin ennustettavuusmittauksiin, työssä tehdään ensin tilastollisia havaintoja. Ennustettavuuden tutkiminen aloitetaan yksinkertaisilla ennustusmalleilla, jotka toimivat nopeasti ja luotettavasti. Konvoluutioneuroverkkojen tapauksessa huomataan, että ne ovat selvästi hitaampia ja hankalampia opettaa, mutta toisaalta toimiessaan tuottavat parhaimmat ennustustulokset. Yksinkertaisemmatkin mallit pystyvät ennustamaan jännitysvoima-venymä—käyriä melko tarkasti, eivätkä häviä konvoluutioneuroverkkomalleille kovin paljon.

Lopuksi määritellään ongelma, jossa ennustetaan pelkkiä vyöryjä. Neuroverkot kykenevät ennustamaan vain suurpiirteisesti joidenkin vyöryjen esiintymistiheyttä, kun vyöryn koko ja vyöryn käynnistävän jännitysvoiman taso ovat sopivat. Vyöryjen ennustettavuus näyttää tulosten perusteella heikommalta kuin jännitysvoima-venymä—käyrien ennustettavuus, vaikkakin ennustusongelmat on määritelty eri tavoilla.

Avainsanat: plastinen deformaatio, koneoppiminen, laskennallinen materiaalifysiikka, dislokaatiot

TABLE OF CONTENTS

ABBREVIATIONS AND SYMBOLS

a	a general scalar
A	activation operator
b	Burgers vector magnitude
$\vec{\boldsymbol{b}}$	Burgers vector
$\boldsymbol{b}$	bias vector
b_k	bias (element of a bias vector)
B	bias operator (translation operator)
c	correlation coefficient
C	convolution operator
Conv	convolutional (neural) network
d	depth (data array dimension)
d_p	pth pinning point's energy depth
D	differential operator
e	the Euler's number ≈ 2.718
E	pinning energy field
f_p	pth pinning point's force field
f_{ReLU}	rectified linear unit (*ReLU*) function
F	pinning force field
h	height (data array dimension)
i	a general iterative index
j	an iterative index over each dislocation
J	artificial density field of relaxed dislocations
k	a general iterative index, running from 1 to K
K	a positive integer (multipurpose)
L	periodic loop length (expressed in the distance units of the unitless system)
L^1	(Lebesgue) 1-norm
L^2	(Lebesgue) 2-norm (Eucledian norm)
L	linear regression operator (affine operator)
Lasso	L^1-regularized linear regression model (LASSO; Linear Absolute Shrinkage and Selection Operator)
m_{kn}	weight (element of a learnable matrix)
M	learnable matrix
M	matrix operator (linear operator)
n	a general iterative index, running from 1 to N
N	a positive integer (multipurpose)
N_j	number of dislocations
N_p	number of pinning points
N	neural network unit operator (nonlinear regression operator)
NN	(simple artificial) neural network
O	a general operator
p	an iterative index over each pinning point
P	pooling operator
q	cumulated probability, defining a q-quantile
r	distance
R	rearranging operator (reshaping operator)
$\mathbb{R}$	the space of real numbers
$\mathbb{R}^N$	the space of real vectors containing N elements
ReLU	rectified linear unit
s	standard deviation

s_p	pth pinning point's energy standard deviation
S	predictability score
t	time
u	a general axis
$\hat{\boldsymbol{u}}$	a general unit vector
v	variance (when with one index) or covariance (with two indices)
$\boldsymbol{v}$	a general vector
w	width (data array dimension)
x	position, or a general variable
x_j	(relaxed) dislocation positions generally
X_p	pinning point positions generally
$\boldsymbol{x}$	a general vector
y	a general variable
$\boldsymbol{y}$	a general vector
$\boldsymbol{z}$	a general, statistically normalized vector (standardized vector)
Δ	difference
ε	strain (ϵ in some images)
μ	mean
μ^*	shear modulus
π	the mathematical pi ≈ 3.142
σ	stress (external stress, applied stress)
σ_{flow}	flow stress (yield stress, yield strength)
χ	effective viscosity

1. INTRODUCTION

Novel machine learning methods have been spreading like a virus through many research fields during the past decade, providing new kinds of solutions to difficult problems involving estimation and dependency search. The field of material physics is researching if the properties of materials, such as plastic deformation processes, are predictable by simpler methods than through extensive simulations or experiments. At the same time, the deformation processes become better understood, since new dependencies between quantities are revealed.

This work focuses on studying the predictability of *dislocation* systems that by assumptions can be modelled as one-dimensional systems. The dislocations interact with each other through a long-range harmonic interaction force and are also affected by a fixed force landscape generated by attractive *pinning points*. One-dimensional systems are efficient to simulate and to describe, which is useful for predictability research that usually requires generation of large datasets that are used to train the predictive models.

Theoretical background will be discussed in chapter 2. Some important terms related to material physics will be explained. The one-dimensional equation of motion derived by Moretti et al. [1] and shown in 2.1.2 is the source of the simulation results that are analyzed in this work. Some important mathematical tools for statistical analysis are introduced. The correlation coefficient is used as the measure of predictability in this work; its properties and suitability for measuring predictability are explained while giving an understandable geometrical interpretation. Introduction to the used machine learning methods is given in a mathematical style using operator notation with the aim of high understandability and independence from the choice of how the methods are chosen to be applied in practice.

Chapter 3 explains some of the practical problems and choices made related to simulation and analysis. Simulations produce *stress-strain curves* that contain the main quantities to be predicted. Parameter choices are explained so that the results obtained for this work become evaluable and reproducible. Predictive model training processes are explained, noting some of the encountered problems and solutions to them.

Statistics and prediction results are then shown in chapter 4. The predictive models attempt to predict the variations of the stress-strain curves between realizations. Before

this, the variations and other relevant properties are studied statistically. Tests are performed to study the dependency of the stress-strain curve on the initial state and how the *flow stress* is affected by perturbations in the pinning configuration. Different prediction methods, including linear regression, simple neural network and convolutional neural network models, are compared when assessing the predictability of the stress-strain curves. Additionally, statistics and predictability of the critical events, *avalanches*, found within the stress-strain curves are studied. An artificial continuous quantity consisting of avalanche densities within a two-dimensional space is derived from the discrete avalanche events and used as the prediction target.

2. THEORY

This chapter explains theoretical background relevant to this work. It is divided into a physics section 2.1 about the origins of the problem to be studied, a mathematics section 2.2 related to statistical analysis, and a machine learning section 2.3 explaining the predictive models used when studying predictability of the simulated physical systems.

2.1 Physics of Material Deformation

First, 2.1.1 motivates and explains some of the terminology used in this work. Also, examples of recent research articles related to the topic are mentioned. Then, 2.1.2 shows the equation of motion that is used to produce the simulation results of this work.

2.1.1 Background and Motivation

When exposed to external stress, plastic deformation of many solid materials is caused by the movement of *dislocations*, defects (displacements) within the crystalline structure. A dislocation is formed when the bonds between atoms/ions rearrange so that the perfect crystal structure is broken. The defects (lattice distortions) typically form a three-dimensional line consisting of a string of neighboring defects, which collectively are called a dislocation. Plastic deformation phenomena related to dislocations are found mostly in crystalline materials, particularly metals, which bond with relatively weak ionic bonds.

A defect has a direction, which can be expressed with a *Burgers vector* (see Figure 1). A dislocation can be said to have a Burgers vector when viewing an intersection, although the vector may not be consistent along the dislocation line when the line is not straight. Dislocations with opposing Burgers vectors can annihilate each other, locally restoring the original, perfect structure. Dislocations with non-opposing Burgers vectors can form pileups (dislocation assemblies), which are the places where events like sliding (gliding) or fracture can occur. Dislocations move easiest along the Burgers vector's axis.

Dislocations can be classified in many ways. For example, they can be divided into immobile and mobile dislocations. Mobile dislocations can be divided into *edge* and *screw* dislocations, based on the orientation of the Burgers vector relative to the dislocation line (Figure 1). Again, edge dislocation pileups can be classified based on how they are stacked relative to the Burgers vectors [1]. One case is the *low-angle grain boundary* (sometimes *small-angle grain boundary*), shown in Figure 2, where the Burgers vector is perpendicular to the dislocation array. Another case is where the Burgers vector is parallel to the stacking dimension of the dislocations; this is the case studied in this work.

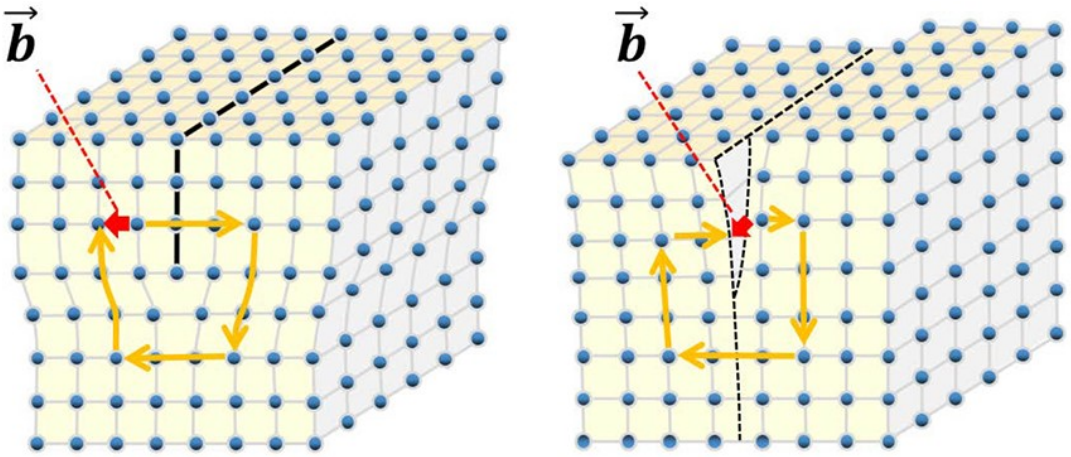

Figure 1. *Schematic pictures of an edge dislocation (left) and a screw dislocation (right) within a cubic crystal lattice. Vectors* $\vec{b}$ *are the corresponding Burgers vectors, defined as the difference in a reference loop (yellow arrows) between the deformed and a perfect structure. Dislocation lines are the strings of points where the displacements initiate, and in this example, continue perpendicular to the yellow (front) face. (An image from [12].)*

In this work, a simplified model of an edge dislocation pileup is studied. The model is derived by Moretti et al. [1]. Dislocations are assumed to be straight lines, which in a pileup form a planar dislocation array that can be effectively modelled as a one-dimensional system, considering only the dimension where the dislocations are stacked. The phenomenon to be studied is the sliding of a pileup as a response to external stress, given a fixed *pinning* landscape (generated by *pinning points*, which represent lattice impurities or other, immobile dislocations) that prevents movement to some extent, like friction. More specifically, this work studies how predictable the response to the external stress is, given that the pinning landscape and the initial state of the dislocation pileup are known.

The dislocations move individually while interacting with each other as well as with the pinning points. External stress can trigger critical events (*avalanches*, also called *slip events*), where one or more dislocations move past pinning obstacles. Eventually, when enough stress is applied, all dislocations of the model start moving collectively, and the material starts to yield. This event is considered (by Moretti et al. [1] and Pun et al. [6]) to be a *second-order phase transition* (*continuous phase transition*), and the corresponding theory suggests certain things, such as a *scaling law* (power law decay) for the occurrence probability of avalanche sizes near the transition threshold, and that the events (avalanches) near phase transition should be unpredictable by nature [6].

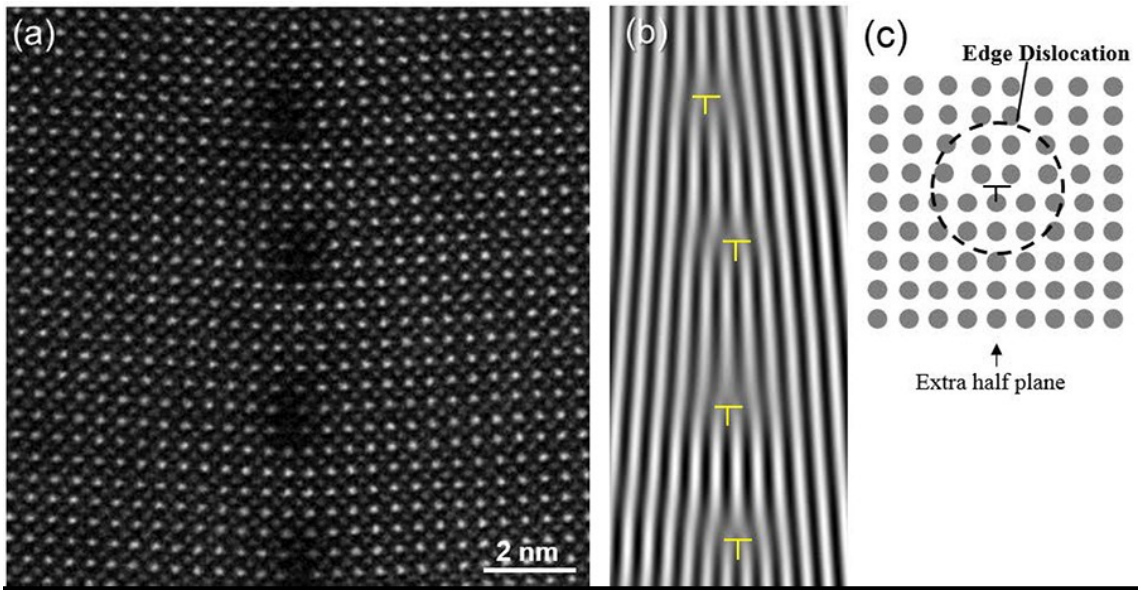

Figure 2. *One case of an edge dislocation array. (a) Low-angle (small-angle) grain boundary in atomic scale. (b) Alternative visualization by connecting vertical planes. (c) An edge dislocation schematic as a remainder. The* T*-symbol is sometimes used [4] as a marker for dislocations, with orientation containing information about the Burgers vector. (An image from [12].)*

As mentioned, this work revolves around studying the one-dimensional system defined by Moretti et al. [1]. As for other work related on the theory of dislocations, Leoni and Zapperi [2] study similar one-dimensional dislocation systems but with mobile, instead of fixed, pinning points (impurities). Sethna et al. [3] review the general theory behind related systems where similar critical events, such as earthquakes, occur, causing *crackling noise*, and try to explain the frequently encountered power law decay of the probability density of critical event sizes.

In some works, machine learning is applied on predicting properties of systems closely related to the system studied in this work. Salmenjoki et al. [4] study the predictability of simulated two-dimensional dislocation systems with two kinds of dislocations (opposing Burgers vectors) without a pinning landscape, applying a simple artificial neural network for predicting stress-strain curves. Miyazawa et al. [5] predict the cyclic stress-strain response (hysteresis) of steels (simulated and experimental), comparing both linear regression and machine learning methods. Pun et al. [6] apply a convolutional neural network to predict critical event sizes in a certain related system (earthquake model).

There are also recent works involving realistic three-dimensional models and experiments where machine learning has been applied, although not always with neural network methods that are used in this work. Steinberger et al. [7] find relations, using machine learning techniques, between microscopic dislocation features (local densities) and the macroscopic structures formed by a large number of dislocations. Pagan et al.

[8] use a certain learning method to predict plastic deformation processes using both experimental and simulated data.

2.1.2 The One-Dimensional Dislocation Model

Moretti et al. [1] derive, after considering general cases in two- and three dimensions, the one-dimensional equation of motion

$$\chi \frac{dx_i}{dt} = \mu^* b^2 \sum_{j \neq i} \frac{1}{x_i - x_j} + b\sigma + \sum_p f_p(x_i) \tag{1}$$

where χ is effective viscosity, x_i is the position of the ith dislocation, t time, μ^* shear modulus, b Burgers vector's magnitude, σ (external, applied) stress, f_p the pth pinning point's force field, chosen to be an attractive Gaussian derivative

$$f_p(x) = -d_p \frac{x - X_p}{s_p^2} e^{-\frac{1}{2}\left(\frac{x - X_p}{s_p}\right)^2}, \tag{2}$$

where $e \approx 2.718$ is Euler's number, X_p is the pth pinning point's position, d_p pinning energy depth and s_p the standard deviation of the Gaussian. (Notice the unnecessary absolute value brackets when viewing the equation corresponding to (1), equation 29, in [1]. Also, equation (2) has been chosen here to be shown slightly differently from the source, matching the parameters with the usual Gaussian parameters.)

The equation (1) is that of a *viscous* (*overdamped*) system, where force causes velocity instead of acceleration as in inertial systems. Lacombe et al. [9] study a closely related system as the one in this work, but with a different, short-range interaction that doesn't extend past immediate neighbor *beads* (representations of dislocations), but in [1] it is explained that a model without long-range interactions is insufficient when describing the movement of dislocations.

For simulations, the expressions are simplified by choosing an *unitless system* where $\chi = \mu^* = b = 1$ and $d_p = s_p^2 = 0.5$. Also, the infinite sums must be finitized. The sum over a dislocation's periodic copies has a finite expression for periodic length L:

$$\sum_{k=-\infty}^{\infty} \frac{1}{x + kL} = \frac{\pi}{L \tan(\pi x / L)}, \tag{3}$$

where $\pi \approx 3.142$ is the mathematical pi. The sum over pinning points is truncated in such a way that only the pinning points that are closer than some cut-off radius R (in this work, $R = 8$) are included, taking advantage of the quick decay of Gaussian functions. All in all,

after slight reorganization and adding brackets for readability, the expression of (1) simplifies to

$$\frac{dx_i}{dt} = \left\{ \sum_{j=1,\ j\neq i}^{N_j} \frac{\pi}{L\tan\left[\frac{\pi(x_i - x_j)}{L}\right]} \right\} + \left(\sum_{p;\ |r_{ip}|<R} -r_{ip} e^{-r_{ip}^2} \right) + \sigma \tag{4}$$

where N_j is the number of dislocations within a unit period and $r_{ip} = x_i - X_p$. Moreover, the average spacing between dislocations is set to $\overline{D_j}$ = 16 and between pinning points to $\overline{D_p}$ = 2. With these, setting N_j is enough to define the periodic system size uniquely: periodic loop length $L = N_j \overline{D_j}$, and the number of pinning points $N_p = L/\overline{D_p}$.

One main product of the simulations are the stress-strain curves. Strain ε is defined as the average displacement of the dislocations:

$$\varepsilon(t) := \frac{1}{N_j} \sum_{j=1}^{N_j} \varepsilon_j(t)\,, \qquad \varepsilon_j(t) = x_j(t) - x_j(0)\,. \tag{5}$$

Simulations are carried out in two phases: a relaxation phase and a main phase. Relaxation phase lets dislocation settle to a (meta)stable configuration without external stress. The stress-strain curve is obtained in the second phase, as the response of the strain to the applied stress. In (5), time is considered 0 at the beginning of the phase in consideration; strain in stress-strain curves is the strain relative to the beginning (relaxed state) of the main phase.

2.2 Mathematics of Statistics

Statistics mostly revolve around *distributions*. Subchapter 2.2.1 explains the terms related to distributions that are relevant to this work. The *correlation coefficient* is widely used in this work, most notably to measure predictability (how well a predictive model works). A way to interpret correlation is introduced in 2.2.2. It is also shown how the correlation coefficient is related to an alternative measure of predictability.

2.2.1 Distributions

Distributions in general are defined as generalized functions, but the main use in this work is for distributions that describe how probability spreads over different possible outcomes of a random variable. These distributions are called *probability density functions/distributions* (for a continuous variable) or *probability mass functions/distributions* (for a discrete variable). Probability mass functions have values that represent probability

by themselves, but probability density functions must be integrated to obtain the probability over the integrated interval.

One commonly encountered distribution is a *normal* (*Gaussian*) distribution, which has a probability density function

$$p(x) = \frac{1}{s\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{s}\right)^2}, \tag{6}$$

where $p(x)$ is the probability density at x, μ the mean, s the standard deviation, $\pi \approx 3.142$ is the mathematical pi, and $e \approx 2.718$ is Euler's number. For a vector $\boldsymbol{x} \in \mathbb{R}^N$ of N realizations (samples) from any distribution, the *sample mean*

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{N}(x_1 + x_2 + \cdots + x_N) \tag{7}$$

where x_i are the elements of $\boldsymbol{x}$. Sample mean is an estimate of the original mean μ.

Standard deviation is a measure of how much the distribution spreads around the mean. A related term, *variance*, refers to squared standard deviation. The *sample standard deviation* for the vector $\boldsymbol{x} \in \mathbb{R}^N$ is

$$s_{\boldsymbol{x}} = \sqrt{\frac{1}{N^*}\sum_{i=1}^{N}(x_i - \bar{x})^2}, \qquad N^* = N - 1\,. \tag{8}$$

N^* can alternatively be equal to N, but only when the true mean μ is known and replaces the sample mean $\bar{x}$ in the equation. The standard deviation of the sample mean (7) is called the *standard error of the mean* and can be calculated as

$$s_{\bar{x}} = \frac{s}{\sqrt{N}} \approx \frac{s_{\boldsymbol{x}}}{\sqrt{N}} \tag{9}$$

where s is the true standard deviation, $s_{\boldsymbol{x}}$ is the sample standard deviation of (8), and N is the number of samples. The standard error of the mean is used to estimate the accuracy of a sample mean, or for evaluating how many samples are needed for reaching enough accuracy.

The integral function of a probability density function, with integration starting from $-\infty$ or from the lowest possible outcome value, is called a *cumulative distribution function*. A *quantile* is the outcome at which the cumulative distribution function reaches a certain value; a *quantile function* is an *inverse function* of a cumulative distribution function (which is always invertible, at least piecewise). In this work, a q-quantile refers to the value of the cumulative distribution function at q, where q is the cumulated probability,

expressed either within the interval [0, 1] or as a percentage within [0%, 100%]. Note that outside this work, there may be other practices for naming quantiles.

Statistical *normalization* (*standardization*) of a vector $\boldsymbol{x}$ is the process where the two statistical properties (mean and standard deviation) are standardized while preserving the relative shape of the signal. In informational sense, often the essential information of a vector is encoded in the relative, order-dependent fluctuation of the vector element values and not in the order-independent properties, such as the mean and scale, of the vector. The statistically normalized version of $\boldsymbol{x}$ is

$$\boldsymbol{z} = \frac{\boldsymbol{x} - \bar{x}}{s_{\boldsymbol{x}}} \tag{10}$$

where $\bar{x}$ is the sample mean (7) and $s_{\boldsymbol{x}}$ the sample standard deviation (8). Note that *normalization* as a term is used for different purposes in different contexts. In the context of mathematical analysis, it usually refers to division by a norm (for example, *normalization* of a count histogram into a probability density/mass distribution).

2.2.2 Correlation

Two ways to calculate and interpret the correlation coefficient are presented. The (*Pearson*) *correlation coefficient* for vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is

$$c_{\boldsymbol{x},\boldsymbol{y}} = \frac{v_{\boldsymbol{x},\boldsymbol{y}}}{s_{\boldsymbol{x}} s_{\boldsymbol{y}}} \tag{11}$$

where $s_{\boldsymbol{x}}$ and $s_{\boldsymbol{y}}$ are the sample standard deviations of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, and

$$v_{\boldsymbol{x},\boldsymbol{y}} = \frac{1}{N^*}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \ , \qquad N^* = N - 1 \tag{12}$$

is the sample covariance of $\boldsymbol{x}$ and $\boldsymbol{y}$. This is the definition often used in the context of statistics. Another way to define the correlation coefficient is to use a functional analysis route, where correlation has a more intuitional geometrical interpretation. This definition requires an *inner-product* and a *norm*. For real vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, an inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ can be defined as the *dot product* $\boldsymbol{x} \cdot \boldsymbol{y}$**;**

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x} \cdot \boldsymbol{y} = \sum\nolimits_i x_i y_i \ , \qquad \text{when } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N, \tag{13}$$

which induces the so-called L^2-norm

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} := \sqrt{\sum\nolimits_i x_i^2} \quad (= \|\boldsymbol{x}\|_2), \qquad \text{when } \boldsymbol{x} \in \mathbb{R}^N. \tag{14}$$

Assuming $\boldsymbol{x}$ and $\boldsymbol{y}$ are both zero-mean vectors, the correlation coefficient can be calculated as the inner product of the *unit vectors* of $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$c(\boldsymbol{x},\boldsymbol{y}) = \langle \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \rangle = \langle \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}, \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|} \rangle = \frac{\langle \boldsymbol{x},\boldsymbol{y} \rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}, \qquad \text{when } \bar{x} = \bar{y} = 0 \tag{15}$$

with the last form analogous with the statistical definition (12), showing that these are the same property. The trick here is that the inner product of any vector $\boldsymbol{v}$ with any unit vector $\hat{\boldsymbol{u}}$ gives the coordinate $v_{\|u}$ along the axis u defined by the unit vector $\hat{\boldsymbol{u}}$ (given any *orthonormal basis* (coordinate system) that includes such axis); $v_{\|u} = \langle \boldsymbol{v}, \hat{\boldsymbol{u}} \rangle$. This is visualized in Figure 3. Another property of the correlation coefficient is that the correlation between random, or independent, vectors is expected to be 0, and the expected deviation from 0 becomes smaller the more elements the vectors have.

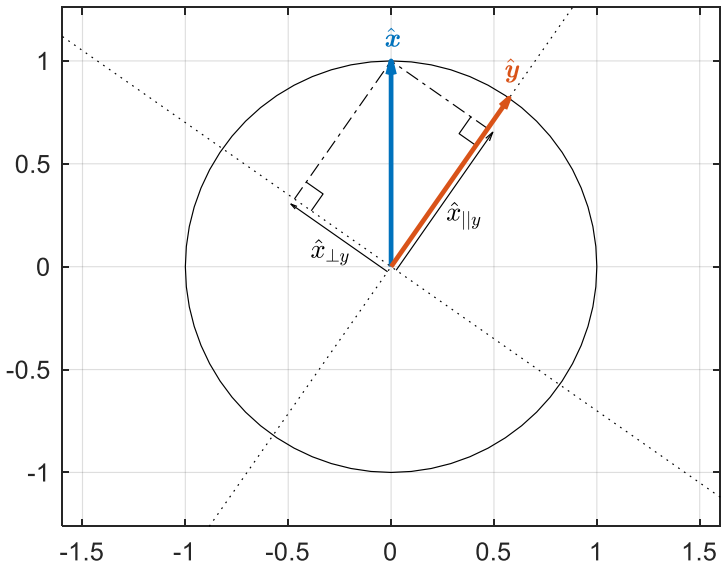

Figure 3. *Illustration of example unit vectors $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ along the unit circle, and of the division of $\hat{\boldsymbol{x}}$ into orthogonal components along ($\|$) and perpendicular to ($\perp$) the axis defined by $\hat{\boldsymbol{y}}$. The original vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ can have any number of elements; the illustration is just within the two-dimensional subspace where the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ reside. If the condition $\bar{x} = \bar{y} = 0$ holds in the original basis for vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, then the correlation coefficient $c(\boldsymbol{x},\boldsymbol{y}) = \langle \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \rangle = \hat{x}_{\|y}$. Note that mean is a property that depends on the choice of basis. Also note that when $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^{\boldsymbol{N}}$, $c(\boldsymbol{x},\boldsymbol{y}) = c(\boldsymbol{y},\boldsymbol{x})$.*

In this work, the correlation coefficient c is used as a measure of how much two vectors resemble each other. There are also other ways to measure similarity between vectors. In [4], a score

$$S(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{\sum_i (x_i - y_i)^2}{\sum_i (x_i - \bar{x})^2} \overset{\bar{x}=0}{\Longrightarrow} S(\boldsymbol{x},\boldsymbol{y}) = 1 - \left(\frac{\|\boldsymbol{x} - \boldsymbol{y}\|}{\|\boldsymbol{x}\|} \right)^2 \tag{16}$$

is used as such measure, where $\boldsymbol{y}$ is interpreted as the estimate of $\boldsymbol{x}$. The error vector $\boldsymbol{x}-\boldsymbol{y}$ normalized by $\|\boldsymbol{x}\|$ would in Figure 3 be a vector from some point along the positive side of axis y (defined by $\hat{\boldsymbol{y}}$) to the tip of $\hat{\boldsymbol{x}}$, with the location of the point on the axis y depending on the relative scale between $\boldsymbol{x}$ and $\boldsymbol{y}$ (the ratio between $\|\boldsymbol{x}\|$ and $\|\boldsymbol{y}\|$). If the scale of $\boldsymbol{y}$ relative to $\boldsymbol{x}$ is optimized so that the error $\|\boldsymbol{x}-\boldsymbol{y}\|$ is minimized, then must be $(\boldsymbol{x}-\boldsymbol{y}) \perp \boldsymbol{y}$, and $\frac{\|\boldsymbol{x}-\boldsymbol{y}\|}{\|\boldsymbol{x}\|}$ equal to $\hat{x}_{\perp y}$ (Figure 3). Then, by the Pythagorean theorem, $S \cong 1-\hat{x}_{\perp y}^2 = \hat{x}_{\parallel y}^2 = c^2$, where the first equality holds when the previously mentioned condition of optimal scale applies. Also, generally applies that $S \leq c^2$. If the optimizer that produces $\boldsymbol{y}$ is assumed to work well enough, then it can also be assumed that $S \approx c^2$. The two measures are therefore comparable, although they are not the exact same property.

2.3 Machine Learning

With advancements in machine learning research, useful machine learning techniques have recently emerged. There are also plenty of terms, such as *artificial intelligence* and *deep learning*, some with varying interpretations. In short, the task which machine learning generally tries to achieve is nonlinear analysis; to learn a nonlinear mapping from some input to some desired output. The tool to achieve this is an optimization method that minimizes difference (measured with *loss*) between model output and desired output. Optimization methods, however, require many conditions to be fulfilled before they can be expected to work desirably, such as *convexity*. Learning a general mapping that would work for inputs outside the learning set usually demands *regularization* techniques which reduce *overlearning* (*overfitting*). In this work, the so-called *L[1]*-regularization method is used, which penalizes for using unnecessary *features* (input elements) and conversely rewarding for concentrating on the most important features (*sparsity*) by adding (scaled) absolute values of learned model parameter values (*weights*) to the loss during optimization.

Modern artificial neural networks form three main branches: *simple artificial neural networks*, which perform nonlinear regression; *convolutional neural networks*, mainly used for image analysis; and *recurrent neural networks* that recreate a memory effect and are mainly used for audio analysis. More generally, networks from these branches can be used as building blocks to create combined models for solving problems that, for example, involve many types of input data. The first two kinds of artificial neural network models are found useful for this work and are introduced in more detail. Input data array

dimensionality depends on the used model, but output arrays, in this work, are always vectors with one or more elements.

For notational purposes, models and their components are presented as mathematical operators. Operators operate here from left to right: $\mathrm{O}x := \mathrm{O}(x)$ and $\mathrm{O}_2\mathrm{O}_1 x := \mathrm{O}_2(\mathrm{O}_1 x)$. This choice has the advantage of having some analogy with the current way of programming neural network models: stacking *layers*. Note that as with other confusing terms, *layer* may refer to different things depending on the context: it can refer to an operation (*convolutional layer*, *activation layer*), to an intermediate result (*input layer*, *hidden layer*, *output layer*) or to both simultaneously.

2.3.1 Simple Artificial Neural Network

These networks are also called *fully connected networks*. A single sample input array has one dimension: *feature* dimension. This type of a model is used [4][5] as an alternative to *linear regression* models (linear fits). Opposed to linear regression, a neural network model has theoretical capability to learn any continuous function (given enough *neurons* (*hidden units*); big enough intermediate result vector), not just linear. In practice, convergence to more complicated than linear shapes is not guaranteed, and some non-linearity learning power is reduced as a side-effect of regularization techniques while preventing overlearning.

First to be defined is the matrix-multiplication (*linear*) operator M:

$$\mathrm{M}\boldsymbol{x} := \boldsymbol{M}\boldsymbol{x} = \begin{bmatrix} m_{11} & \cdots & m_{1N} \\ \vdots & \ddots & \vdots \\ m_{K1} & \cdots & m_{KN} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \tag{17}$$

where m_{kn}, elements of the matrix $\boldsymbol{M}$, are learnable model parameters (*weights*) and x_n are elements (*features* or *descriptors*) of the vector $\boldsymbol{x}$, which is the model input or an intermediate result vector. The biasing (*translation*) operator B is defined with

$$\mathrm{B}\boldsymbol{x} := \boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_K \end{bmatrix}, \tag{18}$$

where b_k are learnable. Combined, these operators form a linear regression (*affine*) operator L:

$$\mathrm{L} := \mathrm{BM}\,, \tag{19}$$

which is the mapping model that a linear regression algorithm optimizes.

Moving to artificial neural networks, there is the addition of a nonlinearity, called *activation*, motivated by the working principle of true neural cells. For this purpose, the biasing operator B is replaced by an activation operator A:

$$\mathrm{A}\boldsymbol{x} := f(\boldsymbol{x} + \boldsymbol{b}) , \tag{20}$$

where f is some function that performs an $\mathbb{R} \to \mathbb{R}$ operation element-wise, and $\boldsymbol{b}$ contains learnable biases. Choices for f include the *identity function* (making A effectively B), smooth versions of the *step function* (such as *sigmoid*; used at the end of some models to suppress output values to a specific range) and the *rectified linear unit* (*ReLU*),

$$f_{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} , \tag{21}$$

which is commonly [4][5][6][10][11] used at intermediate stages of networks. Now, combining A with M produces the neural network unit operator

$$\mathrm{N} := \mathrm{AM} . \tag{22}$$

The operator N itself does not have enough expressive power to recreate any possible continuous function. Operator M can learn to combine input features and scale the resulting combination, and A can apply the nonlinearity with learnable bias. Together, they can recreate functions that resemble the selected nonlinearity function. To obtain full expressive power, two N operators are stacked:

$$\mathrm{N}_i^2 := \mathrm{N}_2\mathrm{N}_1 = \mathrm{A}_2\mathrm{M}_2\mathrm{A}_1\mathrm{M}_1 . \tag{23}$$

The last nonlinearity in the activation A_2 is usually an identity function (*regression* problems) or a *sigmoid*-like function (*classification* problems), depending on the type of the desired output. If the final activation function is the identity function, the model of (23) can be expressed as LN.

Figure 4 illustrates how this model can theoretically approximate any continuous function. The example of Figure 4 is for a mapping from one input feature to one target feature, but the idea works generally by interpreting that the horizontal axis represents any one-dimensional subspace from the input feature space, and the vertical axis represents one output feature, picked from possibly many.

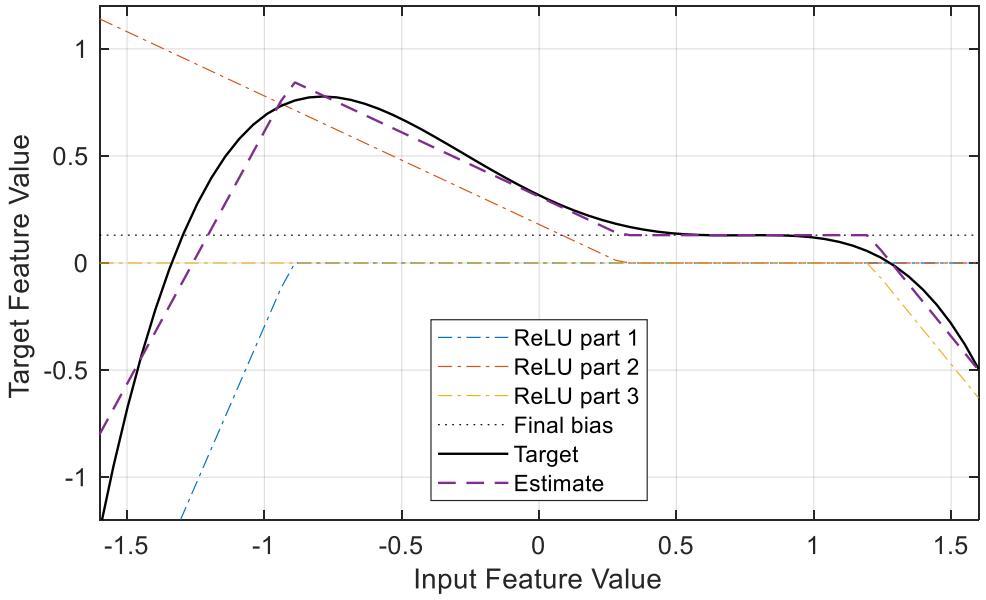

Figure 4. *A handmade example of a target function vs. estimated function, approximated using a simple artificial neural network model,* $\mathrm{LN} = \mathrm{BM}_2\mathrm{AM}_1$*, with 3 hidden units. Each hidden unit produces a ReLU part, which are combined with the final bias to produce the final estimate. A ReLU part is the intermediate result after the multiplying stage but before the summation stage of the* M_2 *operator (*M_1 *contains a 3×1 weight matrix,* M_2 *a 1×3).*

2.3.2 Convolutional Network

Convolutional neural networks operate on data arrays that have *spatial* dimensions along with a feature dimension: a sample input array has one or more spatial dimensions and one feature dimension. For example, (raw) digital images are $h \times w \times d$ -arrays with 2 spatial dimensions (vertical height and horizontal width) and 1 feature dimension (*depth*) that usually has 3 color *channels* and optionally a transparency channel.

Most of the research of convolutional neural networks is performed with images. Krizhevsky et al. [10] study applying a convolutional neural network for classifying image contents in the frequently cited article. Noh et al. [11] show how to apply a convolutional neural network variant for semantic segmentation (producing images showing the areas that each object cover).

As the name suggests, a convolutional network model utilizes convolution operations as a part of the model. The *convolutional operator* C takes some close environment around a *pixel* (spatial element) and performs N_{out} linear combinations on all [6][11] (or sometimes [10] a part of) the feature channels of the environment's pixels to create a new intermediate result pixel with N_{out} features. If the input array has N_{in} feature channels and the convolutional window width is W for all S spatial dimensions, then C effectively performs a matrix multiplication with a learnable matrix of size $N_{out} \times N_{in}W^S$ on a size

$N_{in}W^S$ vector containing the aforementioned convolutional environment. C performs this matrix multiplication process for every input pixel using the same matrix, leaving spatial dimension sizes unchanged after the operation (unless boundary conditions omit pixels that don't have a fully defined environment). This is computationally heavy but, on the other hand, parallelizable. Typically, the convolutional environment is chosen to be small (for example, in this work W = 3; convolutional environment consists only of the center pixel and its neighbors).

Following C in a convolutional network is the activation operator A, defined in (20), which adds a bias for each feature channel before applying the nonlinear function, which is usually ReLU (21). Then, a *pooling operator* P which, after dividing the input array spatially into batches (*pools*), performs some single-valued function on each of the pools, reducing the size of spatial array dimensions. P acts on each feature channel individually but in the same way. Typically [6][10][11], P selects the maximum value from non-overlapping, size 2^S pools (using batching stride 2 for all S spatial dimensions), halving each spatial dimension. In this case, P itself doesn't contain any learnable parameters. In [10], a similar pooling choice, but with slightly overlapping pools, is used.

After $\mathrm{P}_1\mathrm{A}_1\mathrm{C}_1$, a new set of operators, $\mathrm{P}_2\mathrm{A}_2\mathrm{C}_2$ with their own learnable parameters, operates on the intermediate result. This cycle is repeated K times; $\mathrm{P}_k\mathrm{A}_k\mathrm{C}_k$, is followed with $\mathrm{P}_{k+1}\mathrm{A}_{k+1}\mathrm{C}_{k+1}$, until operators $\mathrm{P}_K\mathrm{A}_K\mathrm{C}_K$ have operated. Spatial dimension sizes decay exponentially due to the pooling operators P (although sometimes [10][11] not every AC is followed by a functioning P but a P that is chosen to be an identity operator). It is possible to reduce the array to only one pixel; for example, with convolution window width 3 and pooling stride 2, and without identity P operators, when the original input array has for all spatial dimensions the same size 2^K (if convolutional environment is defined for boundary pixels) or $3 * 2^K - 2$ (if omitting boundary pixels during convolution).

In the case where there is only a single pixel left, the features within the leftover pixel are then linearly combined, biased and possibly nonlinearly activated (if needed for a specific output type) using N, defined in (22), for the final output vector from the convolutional network. More generally, a *rearranging operator* R may be needed before the final N operation. If the intermediate result array after $\mathrm{P}_K\mathrm{A}_K\mathrm{C}_K$ still has more than one pixel, R performs some operation that results in a vector suitable as input for N. In the simplest case, R simply rearranges the array elements into a vector and doesn't contain learnable parameters. In practice, R might be technically required even for the single pixel case to remove singular dimensions if they cause technical problems.

All in all, the *convolutional network* can be described with the stacked operators

$$\mathrm{NR}(\mathrm{P}_i\mathrm{A}_i\mathrm{C}_i)^K = \mathrm{NRP}_K\mathrm{A}_K\mathrm{C}_K \cdots \mathrm{P}_2\mathrm{A}_2\mathrm{C}_2\mathrm{P}_1\mathrm{A}_1\mathrm{C}_1 \tag{24}$$

where C, P and R are described above, A defined in (20), N in (22), K is an integer and i a recursive index. The working principle of the network could be interpreted so that a convolutional operator C seeks spatial patterns, activation A interprets the convolution results (0 for *no* and a positive value for *yes* with value indicating how big *yes*), and pooling P condenses the information while forgetting details about exact position. Fractal repetition scans through different spatial scales giving intolerance to the initial choice of spatial resolution. The collective effect of pooling operators P results in partial or full shift tolerance, where the output depends only on the relative spatial positions of the array elements (how they are located relative to each other) and not on the absolute spatial positions (the choice of where the indexing of the pixels starts/ends). In this work, the studied systems are periodic, and it is known beforehand that circularly shifting an input array in the spatial dimension does not change the results.

3. METHODS

Here, most of the technical details behind the results of this work are explained. In 3.1, details about the simulations that produce the stress-strain curve datasets are explained. Details about the statistical analysis of the simulation results and about the training of the predictive models are described in 3.2. More details are presented later in this work alongside the results in chapter 4. Throughout the work, an unitless unit system (explained in 2.1.2) is used.

3.1 Simulation

Datasets of 10 000 realizations each (differing by the randomly generated pinning configurations) are created using the differential equation (4) for periodic system sizes of N_j = 4, 8, 16, 32, 64 and 128 dislocations. Simulations are performed using MATLAB-software with varying versions (2018a, 2018b, 2019a). Smaller systems' datasets are calculated with a normal desktop computer, while datasets for some larger systems are calculated using a computer cluster service.

A simulation consists of two parts. In the first part (relaxation), the dislocations are first placed uniformly with interval $\overline{D_j}$ = 16, starting from position x = 0 (whether the choice of the starting positions matter is studied in 4.2.1), and then simulated for a fixed time of 4 000 time units. Figure 5 shows relaxation time distribution percentiles. This relaxation part is technically solved using MATLAB's *ode15s*-solver, which has a highly adaptive time step. It is found to be very efficient, since it can detect when the system freezes at the end stage of relaxation and increase the time step so as not to spend any more computational resources than necessary. It is important to set a low enough relative tolerance parameter for all MATLAB's differential equation solvers; the parameter value is set to $1 * 10^{-8}$ for the simulations of this work.

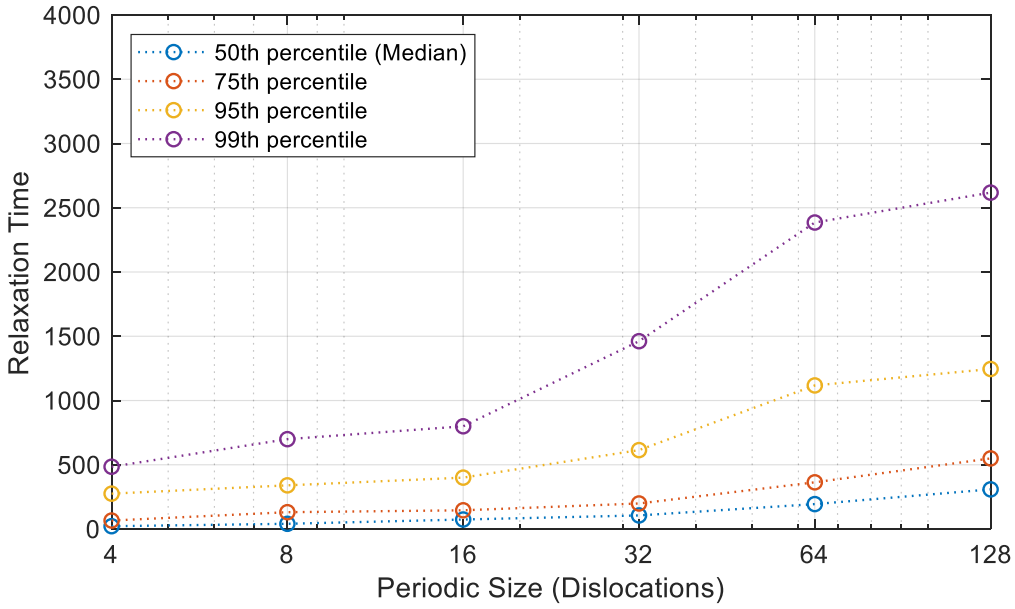

Figure 5. *Relaxation time distribution percentiles (quantiles corresponding to percentages) for different periodic system sizes, calculated from datasets of 1 000 realizations (with varying pinning positions). Relaxation time is the last time point of the relaxation stage where the avalanche condition (to be explained along with equation (25)) is satisfied. In this work, relaxation stage is chosen to end at the fixed time of 4 000 time units.*

An example pinning force and energy (negative integral of the force) are shown in Figure 6. The energy part of Figure 6 also shows relaxed dislocation positions, which are seen to settle at local pinning energy minimums. This is not a general rule however, since the dislocations interact with each other as well, but the interaction effect is small when the dislocations do not get too close to each other as in the case of Figure 6.

As explained in 2.1.2, pinning point positions are derived from a one-dimensional uniform distribution, and all pinning points generate a force function with the same shape, height and width. When looking at the pinning force distribution of Figure 7, which can be calculated from the histogram of the discrete force points (dot markers in Figure 6) of the discretized force landscapes, some force values clearly stand out by forming peaks. The peak at F = 0 is caused by flat regions without nearby pinning points (such as around x = 30 in Figure 6). Other peaks are formed at $F \approx \pm 0.43$, which are the maximum and minimum of a single pinning point's force function. Values near the limits are accentuated in the force distribution since the basic force shape has blunt extrema, producing near-extrema values more frequently than intermediate values. Later in this work, in 4.1, it will be seen that many curves and distributions are affected by the force distribution peaks at $F \approx \pm 0.43$.

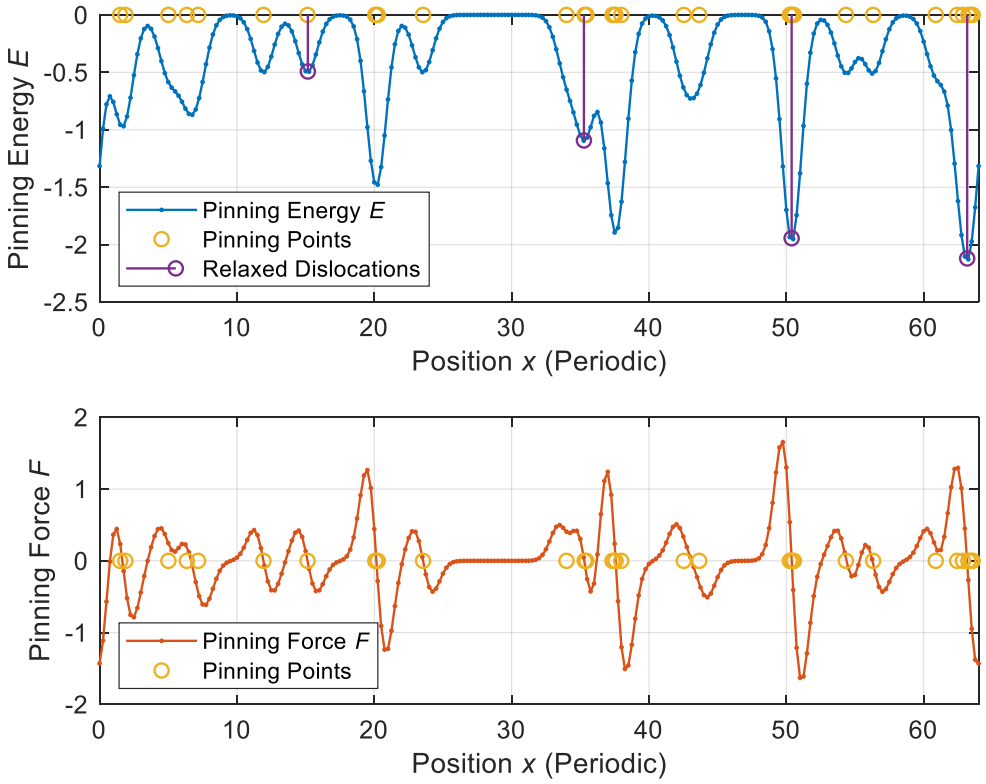

Figure 6. *Example pinning energy (above) and pinning force (below) curves depending on location (periodic system size of 4 dislocations). Pinning points that generate the curves are shown with the circular markers. The top image also shows the dislocation positions after the relaxation stage. (This example is actually the one corresponding to the highest flow stress of the dataset.)*

The second part of the simulation (after relaxation) is the main part where the stress-strain curve is calculated. For a fixed time of 30 000 time units, external stress σ is gradually increased with slope $d\sigma/dt$ depending on the average dislocation velocity $\overline{v_j}$ by the (Fermi-Dirac) relation

$$\frac{d\sigma}{dt} = \frac{m_\sigma}{e^{\frac{r[\overline{v_j}(t)-v_L]}{v_L}} + 1}, \tag{25}$$

where m_σ = 1∗10^{-4} is the maximum limit of the slope, v_L = 2∗10^{-4} is a cut-off velocity value and r = 100 is a shape parameter. This function is a used as a smoother version of a step function; effectively, $d\sigma/dt \approx m_\sigma$ when $\overline{v_j} < v_L$, and $d\sigma/dt \approx 0$ when $\overline{v_j} > v_L$. The choice of using (25) is not intended to affect the resulting stress-strain curves but to add stability for some differential equation solver algorithms that might have trouble with infinitely fast changes in values due to adaptive time steps. Alternatively, the rate of stress could be chosen to depend on the maximum dislocation velocity instead of the average velocity, but the latter choice is used in this work.

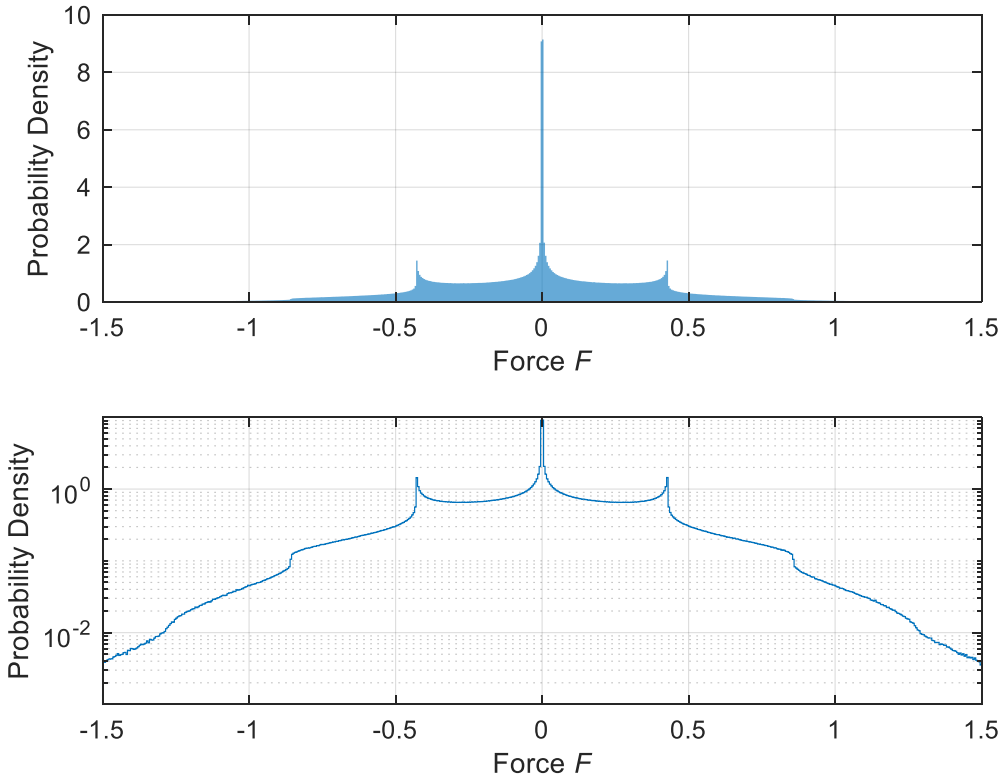

Figure 7. *Pinning force probability density distribution calculated from the dataset of the periodic system size of 64 dislocations, with linear (above) and logarithmic (below) vertical axis choices.*

The differential equation of (25) is provided to the solver along with the actual equation of motion (4). This second simulation part is technically solved with MATLAB's *ode113*-solver with relative tolerance parameter set to $1*10^{-8}$. This solver is found to be more efficient, but not with big margin, than other *non-stiff* problem solvers such as *ode45*, while the *stiff* solvers, such as *ode15s*, are found to be clearly less efficient. Simulation results are saved with a fixed time step Δt = 20 (not with the adaptive time step used by the differential equation solver).

Examples of stress-strain curves are shown in Figure 8 for different system sizes. Figure 9 shows the individual movement of each dislocation for an example system. The stress-strain curves are monotonically increasing functions typically consisting of two kinds of pieces: nearly vertical sections, where the dislocations stay almost put and external stress σ is increased, and horizontal sections (avalanches), where the dislocations move while the external stress σ is held still until the avalanche ends. Dislocations move slightly outside avalanches; this is called *elastic* deformation, while avalanches correspond to *plastic* deformation. Note that depending on the studied system, the stress-

strain curve may be better visualized with either linear or logarithmic axis choice. In this work, strain axes are chosen to be logarithmic.

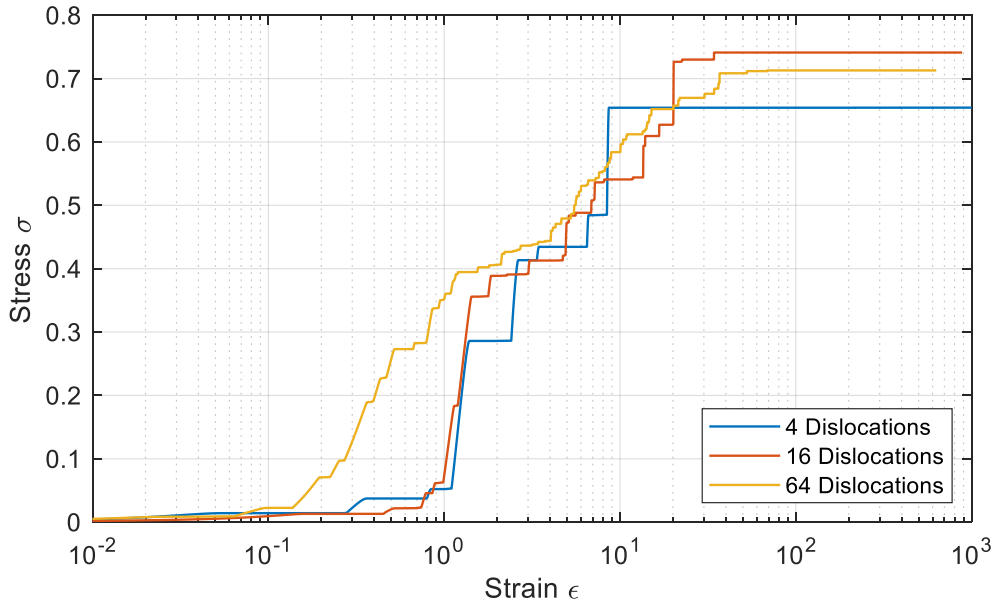

Figure 8. *Example stress-strain curves from different systems with different periodic sizes. Horizontal sections (avalanches) get smaller and their numbers increase as the system size is increased. Here, the curves have an end point since simulation time is finite; other Figures in this work show only such strain intervals where all stress-strain curves of the dataset have data within.*

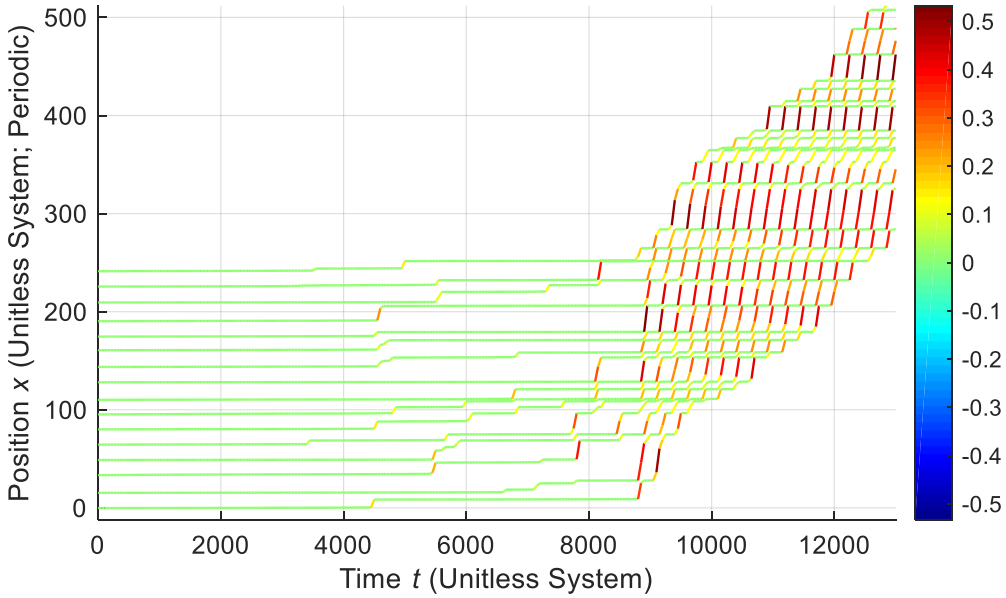

Figure 9. *Time evolution of the position of each unique dislocation of an example periodic system of 16 dislocations. The color corresponds to velocity of the dislocation. All dislocations have an infinite amount of periodic copies with intervals L = 256, but for simplicity, the copies are not shown. Around $t \approx 9\,000$, the system reaches flow stress. Dislocation movements before the flow state correspond to avalanches.*

Avalanches are technically defined as the streaks of saved points where $\Delta\sigma/\Delta t < m_\sigma/2$, that is, where (25) is considered to still the stress. Time resolution (Δt = 20) sets the resolution limit for the avalanche statistics. More specifically, avalanches that have a stress difference $\Delta\sigma \geq \Delta t * m_\sigma/2 = 1{*}10^{-3}$ can be detected as separate avalanches. This accuracy is just about enough for the systems studied in this work, but it should be noted that the time resolution should be increased if studying even larger periodic systems.

With high enough stress, an eternal avalanche occurs, and the (smallest) stress at which this happens is the so-called *flow stress* σ_{flow} (also called *yield stress* or *yield strength*). Figure 10 shows distributions of when the flow stress is reached for each system size. Technically, the time is the starting time of the last avalanche if the simulation ends during an avalanche, or the time t = 30 000 (simulation end time) if the simulation doesn't end during an avalanche. As seen in Figure 10, larger systems require longer simulation time for reaching the flow stress. This is due to increasing number of avalanches; an increasing amount of simulation time is spent at avalanches. If datasets were to be calculated for even larger systems than N_j = 128, the simulation time may not be long enough, but as shown in Figure 10, the chosen time of 30 000 time units is enough for the system sizes studied in this work.

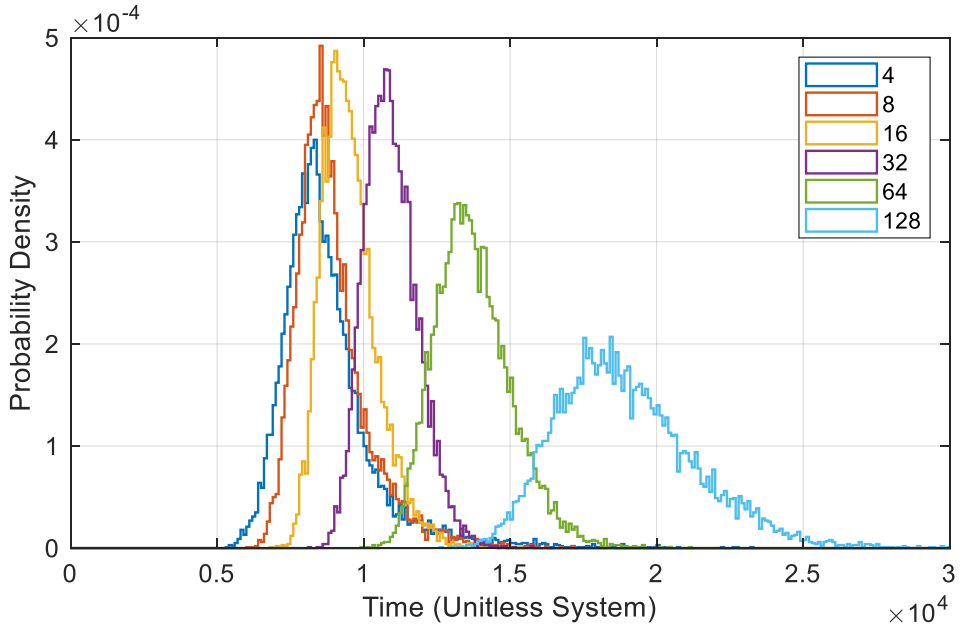

Figure 10. *Probability density distributions of the time at which flow stress is reached for periodic system sizes of 4…128 dislocations. Each distribution is calculated from a dataset of 10 000 realizations. (This result depends on the choice of parameters for equation (25).)*

3.2 Analysis

Analysis of the simulation results is divided to statistical and predictive. Statistical analysis includes estimating probability density distributions of the stress-strain curves, flow stresses and avalanches, finding how changing the periodic system size affects the simulation results and estimating the robustness of the system to certain changes in the initial conditions.

Simulation results are trivially completely predictable since the simulation algorithm is deterministic. The idea in this work is to research whether the results follow some simpler patterns that don't require the use of the simulation algorithm, but could be estimated from the essential system parameters (locations of the pinning points, or fields which they generate) and from the initial state (positions of the relaxed dislocations, or pinning field values at the dislocation positions). This work tests two prediction methods: predicting from selected features without explicit spatial information, using either a (regularized) linear regression model *Lasso* or a simple artificial neural network model (2.3.1), and predicting from spatially defined fields, using a convolutional neural network (2.3.2). While most analysis is carried out with the Matlab software, the training of the predictive models is performed using the Keras library/interface on Python programming language.

Datasets, each consisting of 10 000 samples, are divided randomly into a training set (80% of the data) and a testing set (the remaining 20%, not used for the training). All prediction results are results on the testing sets, averaged over 5 training instances, each with different division into training and testing sets (and the training algorithms themselves are not *deterministic* but *stochastic* since they involve the use of random numbers).

The training of a model is in practice an optimization process where a property called *loss* is minimized. Loss is a measure of the error, typically intensive (error per element, when predicting multiple targets), with the addition of a possible penalty/reward from a regularization method. In this work, all models, except most of the convolutional network models, utilize the L^1-regularization method, which adds the absolute values of the learnable weights (multiplicative matrix parameters, not additive bias parameters) to the loss as a penalty with a user-defined parameter (coefficient). Programmatically, this is done with the *kernel regularizer* option. In future, it could be possible to obtain the optimal parameter choice automatically, but this is not the case yet, so the user has to choose the parameter values.

Well-working regularization both prevents overlearning and improves predictability; Figure 11 illustrates the effects of regularization on the loss curves. L^1-regularization also

provides sparsity, enabling a straightforward weight analysis (a learned weight will have magnitude depending on the importance of the corresponding feature). Other regularization methods exist as well; one common [6][10] method for convolutional networks is the *dropout* method (not used in this work nor in [11]), but the working principle is quite different from, if not opposite to, that of the L^1-regularization.

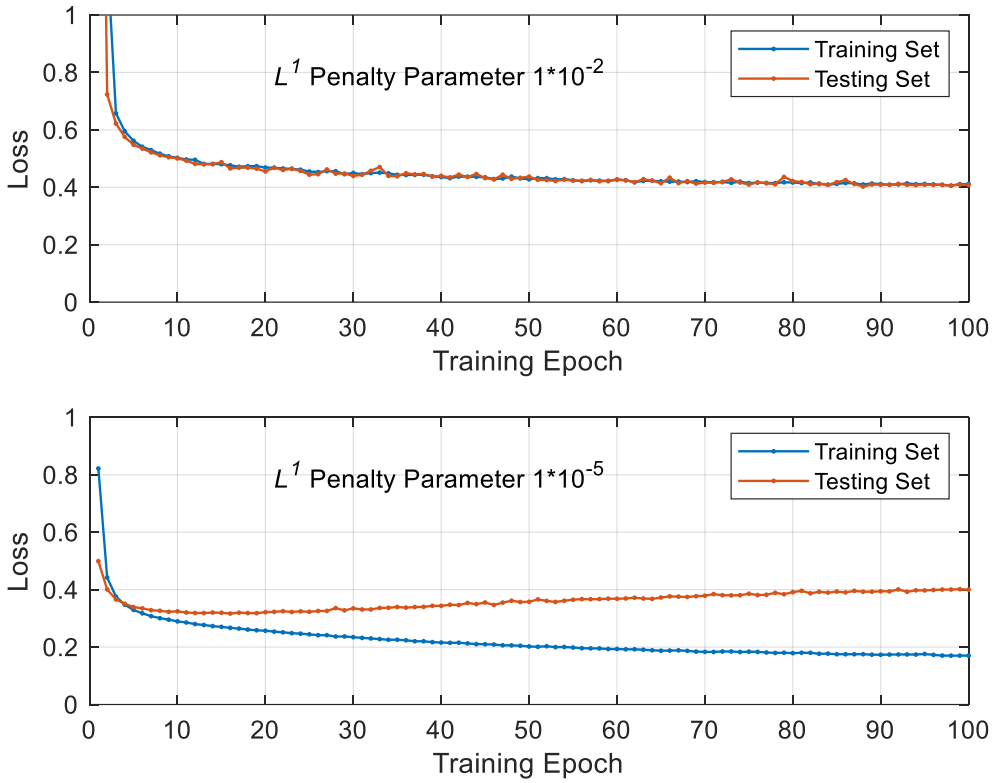

Figure 11. *Example losses of training and testing sets when predicting stress-strain curves of the 64 dislocation system dataset with a simple neural network of 64 hidden units (intermediate result features), using a high (above) or low (below) L^1 penalty parameter. Without enough regularization, overlearning can occur, where the model learns too complicated, non-generally applicable relations, and the training loss diverges from the testing loss.*

Figure 12 shows how the loss curves have a different shape in some cases when training a convolutional network model, causing problems when using a penalty regularization. In this work, convolutional networks don't utilize L^1-regularization, apart from one case, where only flow stress is predicted. However, overlearning is not found to be too big of an issue. This could be because the convolutional window is chosen to be very small (3 pixels; center and its neighbors), preventing overly complicated mappings, and because of the large amount of samples in the datasets.

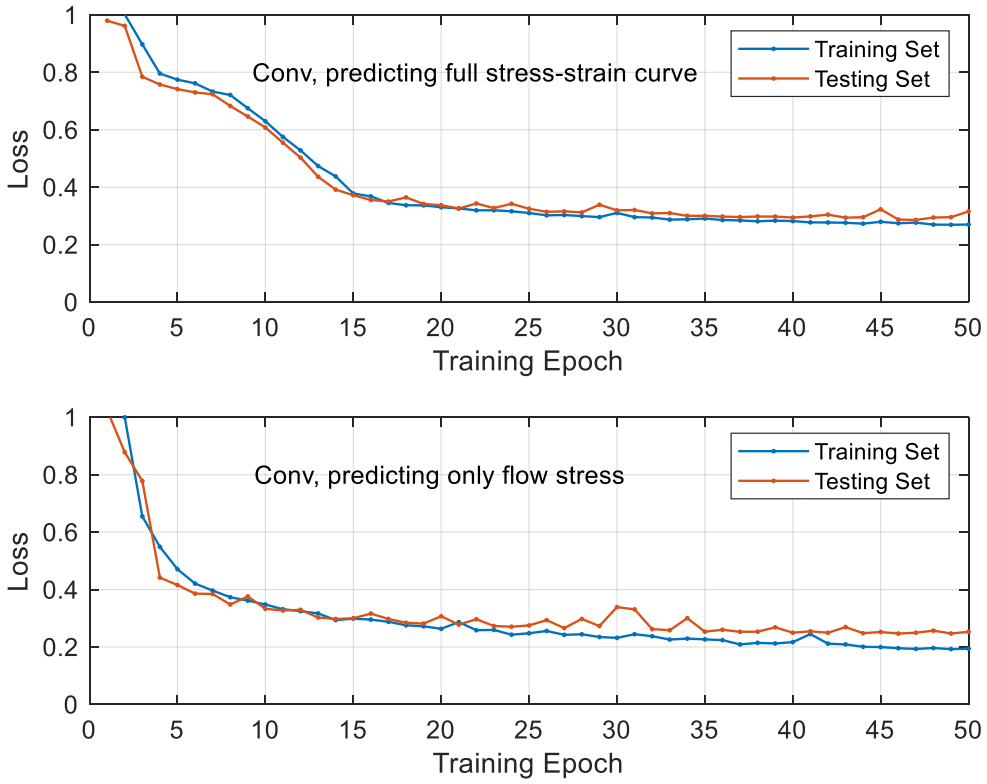

Figure 12. *Example loss curves when predicting stress-strain curves of the 16 dislocation dataset with a convolutional network, focusing on all strain points at once (above), or focusing only on the flow stress (below). The above loss curve has stair-like delays where the optimizer does not immediately find the direction where the loss can be quickly decreased. This prevents the use of penalty regularization methods, since when the loss does not decrease quickly enough, the penalty becomes dominant and suppresses the learnable weights to 0. In the case of focusing only on the flow stress and with the typical exponential decay loss curve shape, L^1 penalty can be used (set to $5*10^{-4}$). Loss curves of single-point predictions of non-flow stresses vary between the two cases.*

Defining the predictive models involves making parameter- and other choices which are shortly explained here. The Lasso model includes a built-in optimizer, while the other models require user-defined optimizer and parameter choices. The optimizer is chosen to be *Adam* with varying learning rate and rate decay. Learning rate is mainly $1*10^{-3}$ (training for 50 or 100 epochs) but sometimes, when the choice does not cause drawbacks, $1*10^{-2}$ (training for 10 epochs). Learning rate decay is either 0 or, in the case of convolutional networks, $1*10^{-5}$. Loss is defined to be the *mean squared error* (with the addition of a possible regularization penalty). Regularization parameter values and the numbers of hidden units (features within intermediate results) for the models predicting from selected features are listed in Table 1. For operators N, the activation function is always chosen to be *ReLU* of equation (21).

Model input and output data is normalized (standardized) individually for each feature channel using equation (10); mean and standard deviation are taken over the samples of a dataset (realization dimension of the dataset array) and, in the case of the convolutional network model inputs, over the spatial dimensions as well (so as not change the relative shape of the spatial fields), producing mean and standard deviation vectors with the number of elements equal to the number of feature channels. If a model fails to learn anything, the result will be the mean, having 0 (or undefined) correlation with the target. Dividing each output (target) feature channel individually by the corresponding standard deviation ensures that, when predicting multiple target features simultaneously, the model focuses on them evenly, not just on the target features with the largest scale variations.

Table 1. *Description of models used for prediction from selected features. The number of hidden units is the number of features (here elements, generally channels) in the intermediate result (change in dimensionality is a consequence of matrix operators* M_i*). The fifth model has 3 intermediate results (hidden layers) having 64 hidden units each. Column 2 shows the equivalent model operator using notation from 2.3.1. Columns 3-6 show the* L^1 *penalty parameter applied to each weight matrix of the models.*

Model	**Operator**	L^1 **(**M_1**)**	L^1 **(**M_2**)**	L^1 **(**M_3**)**	L^1 **(**M_4**)**
Lasso	$L = BM$	$1*10^{-2}$	-	-	-
Neural Network (16 hidden units)	$LN = BM_2AM_1$	$1*10^{-2}$	$1*10^{-5}$	-	-
Neural Network (64 hidden units)	LN	$1*10^{-2}$	$1*10^{-5}$	-	-
Neural Network (64 hidden units) v2	LN	$1*10^{-3}$	$1*10^{-5}$	-	-
Neural Network (3∗64 hidden units)	$LN_3N_2N_1$	$1*10^{-2}$	$1*10^{-5}$	$1*10^{-5}$	$1*10^{-5}$

Inputs for the convolutional network models are discretized versions of continuous functions with discretization step of 0.25 length units of the unitless system (see the dot markers in Figure 6 for a reference). A special feature, J, indicates the relaxed dislocation locations with a function formed by the summation of normal distributions (each representing one dislocation) that, for each dislocation, have mean at the dislocation position and standard deviation of 1 unit of the unitless system.

Using operator notation, all the convolutional models used in this work are equivalent to (24), with a note that the last activation function is the identity function, making the last operator N (22) effectively L (19), just like in the case of the regression models. Convolutional window width is chosen to be 3 pixels, as mentioned before. All pooling operators P are chosen to perform a *max-pooling* operation with non-overlapping pools of size 2. All convolutional network model intermediate results (*hidden layers*) have 16 feature channels (*hidden units*). The number of PAC-cycles, K, is chosen so that the model becomes completely shift-independent. This is done by using the method of reduction to a single pixel that is explained in 2.3.2. In the case of the studied 16 dislocation system, the original inputs have 2^{10} spatial elements, so K is chosen to be 10.

Since the systems studied in this work are periodic, the convolutional operators C should use periodic boundary conditions. In practice, a periodic boundary condition is not yet a built-in option within Keras, so the input data is instead circularly copied to almost triple the data size of the original, and the *valid* boundary condition is used that omits boundary elements during convolution. Operator R is simply a *flatten* operation which doesn't do anything in mathematical sense but resolves a programmatical problem related to array dimensionality.

In the case of avalanche prediction, the outputs (targets) form two-dimensional maps, predicted with a simple artificial neural network of 64 *hidden units*. In practice, a map is provided to the predictive model as a vector of pixels that are interpreted as individual target features. Normalization is therefore carried out individually for each element of the output vector; there are no special *spatial dimensions*, like in the case of convolutional model inputs, that would be treated in a different way than the usual feature dimension is. Also, maybe due to the large number of model outputs, too high regularization parameters are found not to work so well as in the stress prediction cases, so the L^1-regularization parameter is set to $1 * 10^{-5}$ for both weight matrices in the avalanche prediction case.

4. RESULTS

In this chapter are presented the statistics of the simulated stress-strain curves (4.1) and of the avalanches they contain (4.5). Two tests (4.2) show how much the stress-strain curves depend on the dislocation initial state (4.2.1), and to what extent the pinning points can be repositioned without affecting flow stress (4.2.2). Prediction of the stress-strain curves in 4.4 is carried out either from hand-picked features by simple regression models (4.4.1) or from discretized, spatially defined functions by convolutional neural network models (4.4.2). The hand-picked features end up being *quantiles*, as explained in 4.3. Lastly, the same hand-picked features are used to predict avalanche locations in a bivariate map, predicting stress and avalanche size simultaneously (4.6). An unitless unit system is used throughout the work, as explained in 2.1.2.

4.1 Stress-Strain Curve Statistics

Figure 13 shows distributions formed by sets of 10 000 stress-strain curves each for periodic systems having 8, 16, 32 or 64 dislocations. There is a noticeable bend after stress $\sigma \approx 0.4$, most likely caused by the peaks in the pinning force distribution (Figure 7). Flow stress seems normal distributed, as seen in Figure 14, but this does not apply generally for the stress distributions at non-flowing states.

More statistics related to Figure 13 and Figure 14 are presented in Figure 15. As the periodic size increases, the average stress-strain curve converges to a characteristic shape. Reaching flow stress requires higher strain the larger the system size is. The hypothetical stress-strain curve of an infinite system may never reach flow stress but instead saturate towards some limit stress value.

Flow stress distributions have a roughly constant mean that slightly decreases with increasing system size. The standard deviation decays with relation $s_{\sigma_{flow}} \sim 1/\sqrt{N_j}$, where N_j is the number of dislocations. This is the same relation as for the standard error of mean in equation (9) and therefore quite expectable.

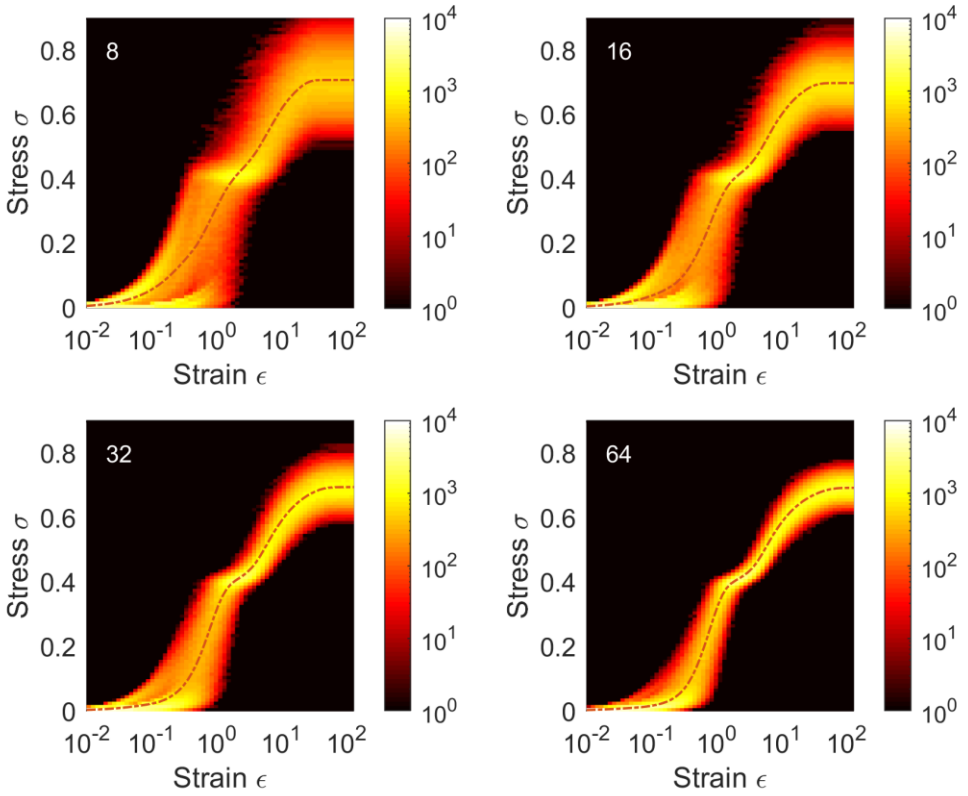

Figure 13. *Stress-strain curve histograms for periodic system sizes of 8, 16, 32 and 64 dislocations, size indicated by the white number at the corner of each image. Dash-dotted lines show the average stress-strain curves.*

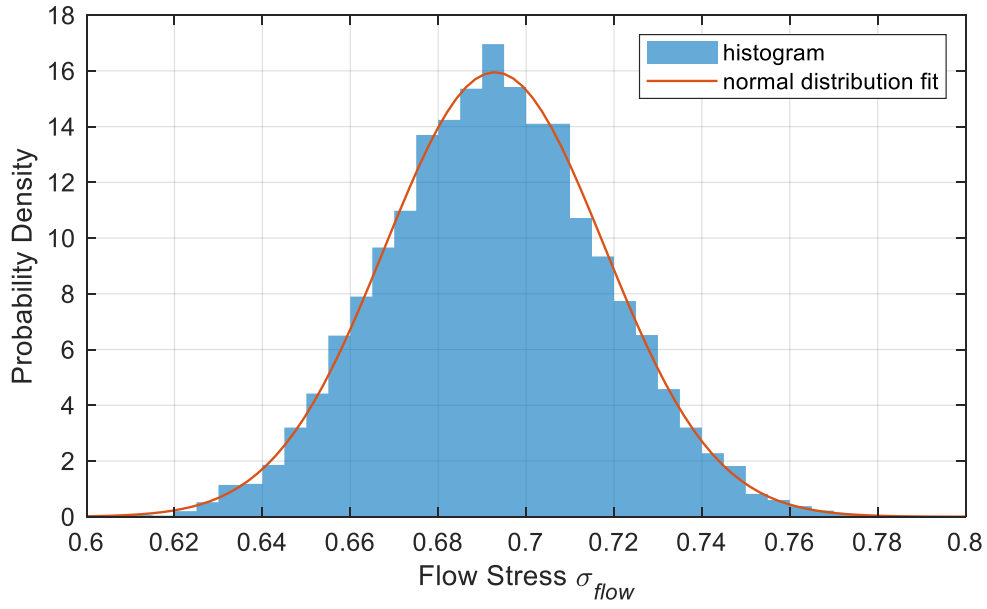

Figure 14. *Probability density distribution of flow stress, obtained as the normalized histogram of the 64 dislocation dataset using bin width 0.005, and the corresponding normal distribution fit, made using the sample mean and sample standard deviation of the flow stresses.*

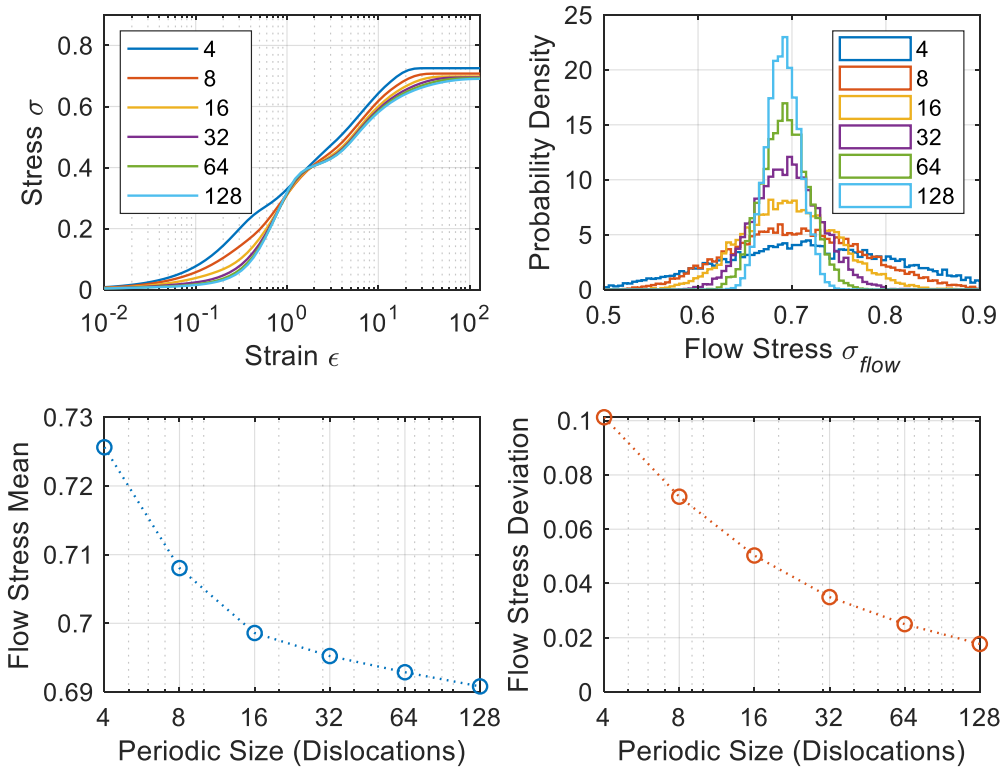

Figure 15. *(Top-left:) Average stress-strain curves for periodic system sizes of 4…128 dislocations. (Top-right:) Flow stress probability density distributions for the different system sizes. (Bottom row:) Flow stress distribution sample mean (left) and sample standard deviation (right) depending on the periodic system size.*

Whether there is a connection between flow stress and the rest of a stress-strain curve (does a high/low flow stress indicate high/low stress values in general), Figure 16 tries to answer this by showing the correlation of stress-strain curves with the corresponding flow stresses. At high strains, stress values contain some indication of the scale of the upcoming flow stress, and the correlation increases with strain. The beginning of the stress-strain curve (below strain $\varepsilon = 1 = 10^0$) seems to be independent of the relative flow stress level, apart from the very smallest systems. An explanation for the smallest systems could be that they produce flow stresses with larger standard deviation (Figure 15), and the beginning of a stress-strain curve may slightly correlate with the flow stress if the flow stress is small enough. This effect then disappears in larger systems where the minimum limit of flow stress increases.

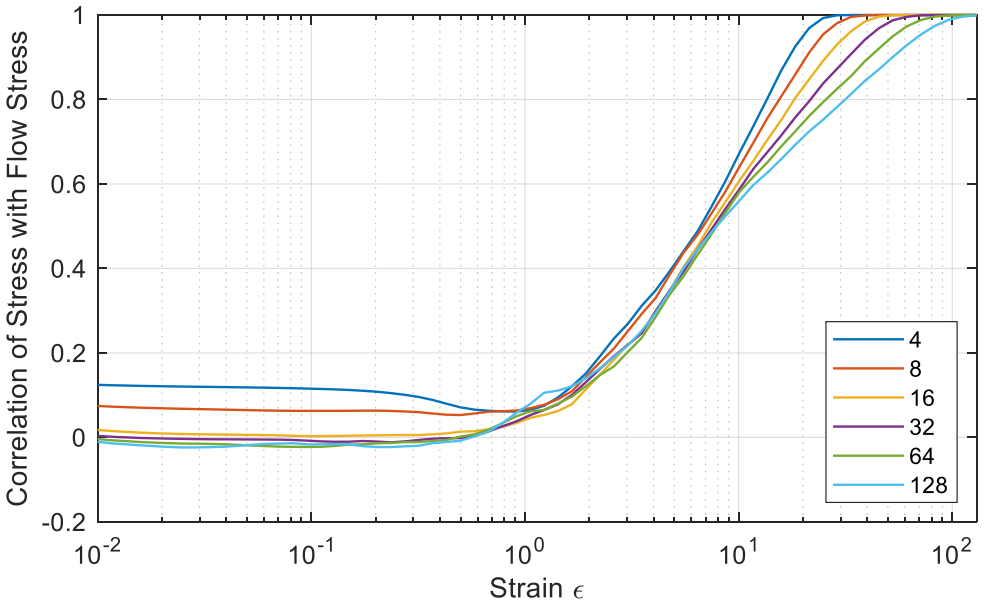

Figure 16. *Correlation between flow stress and stresses at given strains for periodic system sizes of 4…128 dislocations.*

4.2 Tests of Robustness

Two tests are performed to test how much a stress-strain curve depends on the initial dislocation configuration (4.2.1) and how flow stress changes when pinning points are slightly moved from a reference configuration (4.2.2). In both cases, system size is fixed to 64 dislocations. The test datasets again consist of 10 000 realizations each but are specially tailored for the individual purposes of each test. Simulation time (after relaxation) is 20 000 time units for these two tests, since it is found to be enough for the system size of 64 dislocations (see Figure 10).

4.2.1 Initial State Perturbation Test

First to be studied are effects caused by using different initial dislocation positions with fixed pinning positions. A set of 100 different pinning configurations is generated. For each pinning configuration, 100 random shifts within the range [0, 16) (because initially, distance between neighboring dislocations is 16) are drawn from a uniform distribution. The shift is added to all the default (x_j = 0, 16, 32, …) initial dislocation positions (which remain uniformly spaced before the relaxation stage). Combinations of a pinning configuration with a shift for the initial dislocations create a dataset of size 100x100=10 000 in total.

For each stride, containing the 100 different initial conditions for a fixed pinning configuration, the standard deviation of the stress-strain curves at given strains are calculated.

Taking the mean over all 100 strides (pinning configurations) of those standard deviations produces the curves seen in Figure 17 (left side). The resulting deviations can further be compared to the mean stress at each strain, giving a sense of the relative scale (Figure 17, right side).

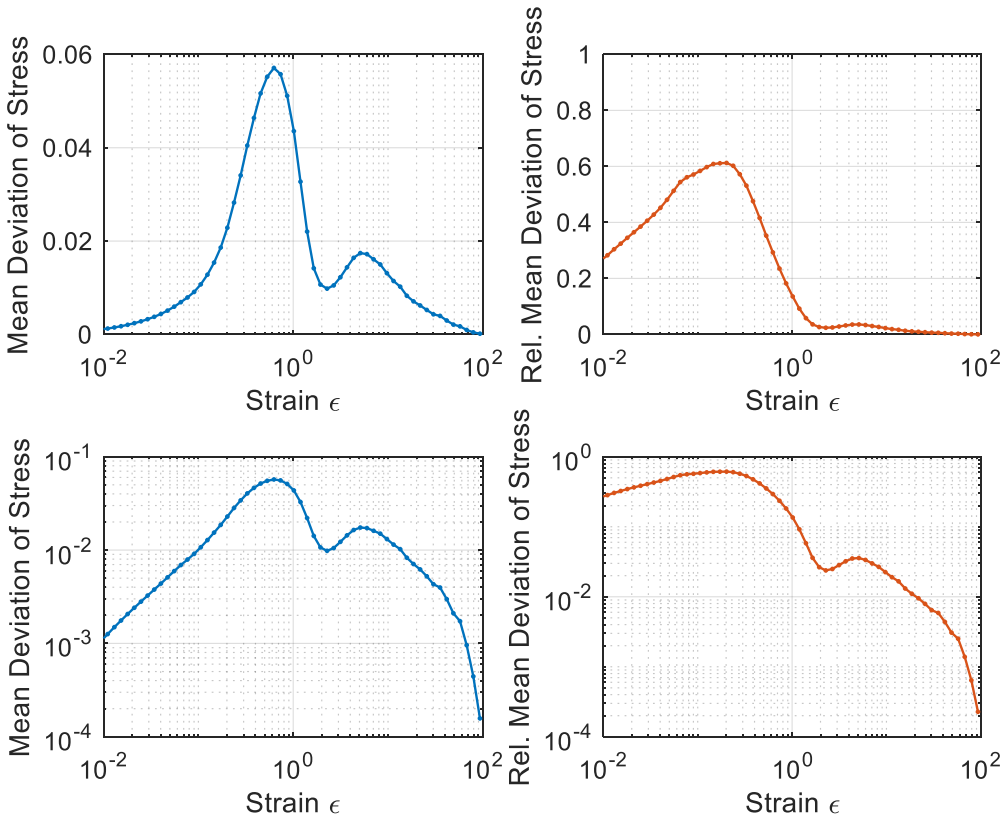

Figure 17. *(Left:) Sample mean over strides (pinning configurations) of sample standard deviations of stress as a function of strain. (Right:) A relative version, obtained by dividing the curve on the left by the mean stress (dash-dotted line at the lower-right in Figure 13). Top row shows each curve with a linear vertical axis, and the bottom row with a logarithmic vertical axis.*

While the choice of initial dislocation positions affects the initial parts of the stress-strain curve quite much, flow stress seems to be very tolerant to such choice. Systems with different initial dislocation configurations but with same pinning configurations produce flow stress sets with no notable deviation. The test suggests that flow stress depends solely on the pinning configuration, not on the initial state of the dislocations.

4.2.2 Pinning Point Perturbation Test

For the second test, the initial dislocation positions are fixed while the pinning positions are perturbed. Like in the previous test, 100 pinning configurations are first generated. These are used as reference configurations which are to be perturbated. For each reference configuration, 100 values of standard deviations each define a zero-mean normal

distribution from which random shifts are drawn for each pinning position separately. One case in each stride has standard deviation 0 (no shifts), acting as the reference case, to which the remaining 99 cases are compared.

The chosen standard deviations can be interpreted as predetermined standard errors of pinning positions. In Figure 18 are shown the effects of pinning position perturbation on flow stress. The top part shows the error (deviation from reference) of flow stress relative to the theoretical saturation value, which is obtained from the standard deviation of the flow stress (~0.025, as seen in Figure 15, bottom-right) by multiplying it with $\sqrt{2}$. This is done since the error between two uncorrelated random variables is a random number from a distribution with variance that is the sum of the component variables' variances.

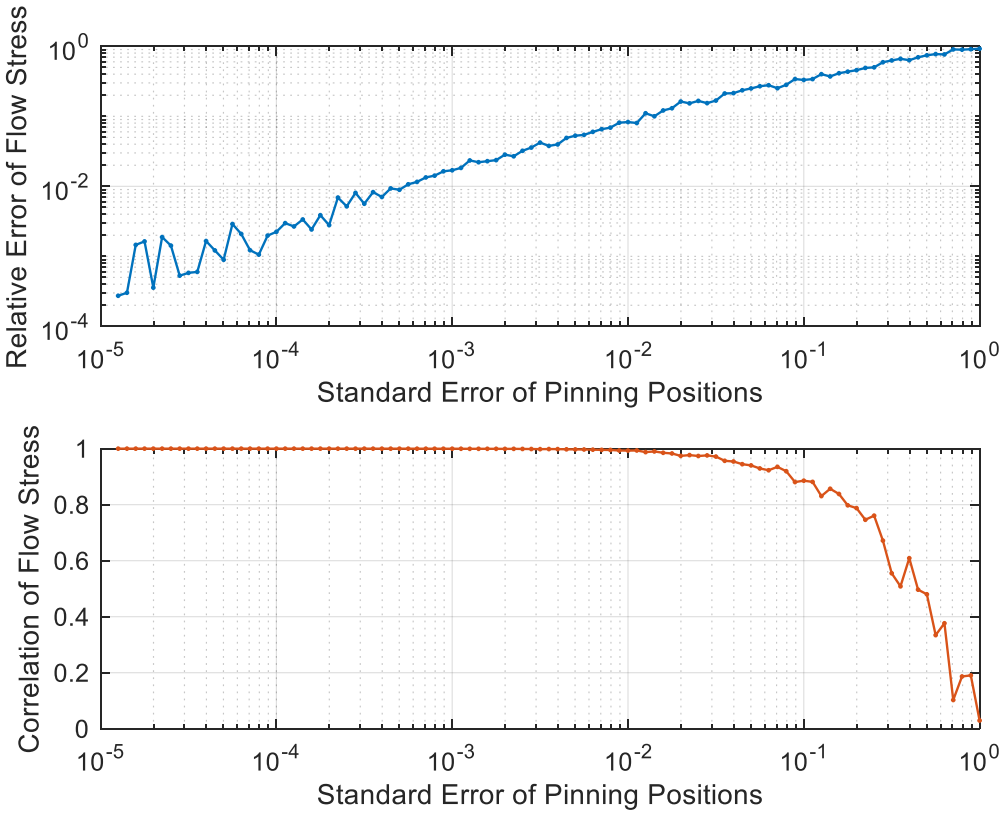

Figure 18. *(Above:) Standard error between perturbed system flow stress and the reference system flow stress, divided by the theoretical saturation value of the error. Here, 1 (= 10^0) on vertical axis suggests full uncorrelation. (Below:) Correlation coefficient of perturbed system flow stress with the counterpart of the reference case.*

The results show steady increase of error in the flow stress before reaching saturation level at high perturbations. The drop in correlation coefficient can be used to determine how accurately the pinning positions need to be measured or given to a predictor if a given accuracy is wanted for the evaluation of the flow stress from the pinning positions.

4.3 Features for Prediction

The predictability research section starts with finding suitable features from which to predict. A simple artificial neural network model (and a linear regression model) requires features expressible with a single scalar number, while the convolutional network model handles feature channels with spatial variation. The predictive value of a feature can be estimated with the correlation coefficient (2.2.2), calculated between the feature and the target to be predicted.

Starting with prediction of flow stress, it could be assumed to depend on the pinning obstacles the dislocations have to overcome. These can be numerically expressed by, for example, finding the location where the pinning force opposing the applied stress is the strongest. It turns out that the system does not depend solely on a single pinning field value, unlike it could be expected for a simpler system without interactions between dislocations. For example, the second most negative pinning force (taken from a discretized pinning force) field gives higher correlation magnitude with the flow stress than the most negative pinning force. Carrying on with the next values from the sorted list of the pinning forces (or other fields) and switching sorting indices (*ranks*) to relative values between 0 (lowest value) and 1 (highest value) leads to the quantiles, explained in 2.2.1.

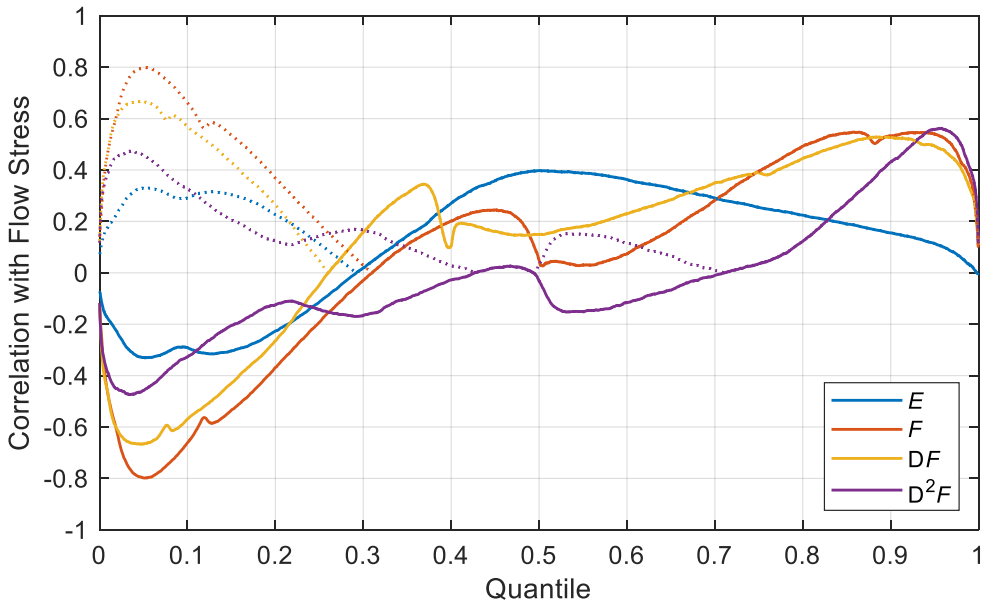

Figure 19. *Correlation between flow stress and quantiles of pinning potential energy (negative integral of pinning force) E, pinning force F, force derivative* D*F and force second derivative* D^2*F distributions for a periodic system of 64 dislocations. Dotted lines show the absolute value of the correlation for easier comparison of magnitudes.*

Figure 19 shows the correlation coefficient between various quantiles and the flow stresses from the dataset of the 64 dislocation system. The 5%-quantile of the pinning

force seems to give the highest (anti)correlation magnitude, around (-)0.80. Figure 20 shows how the force quantile curve depends (not) on the periodic system size.

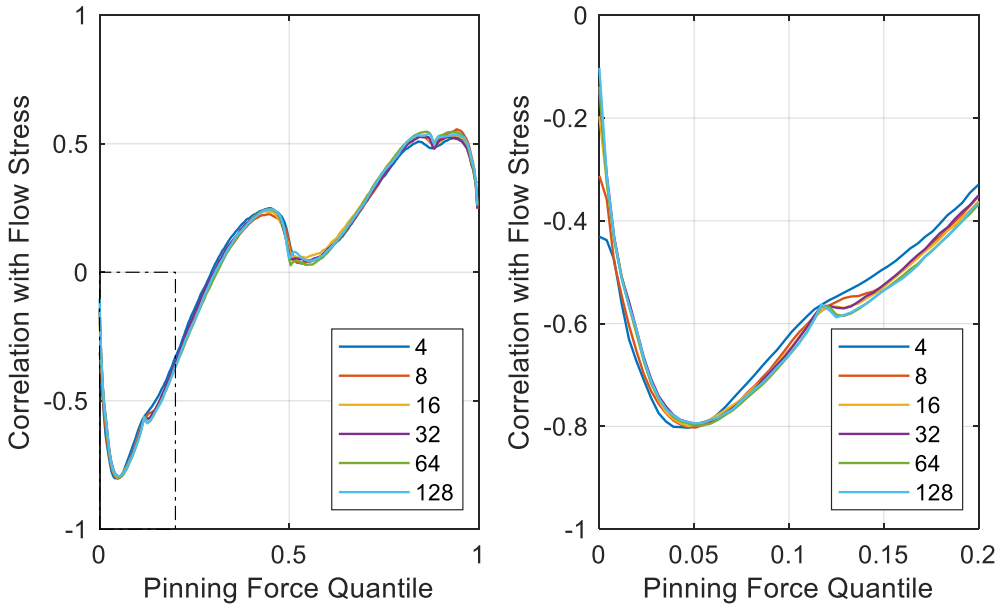

Figure 20. *Flow stress correlation with pinning force quantiles for different periodicity sizes, varying from 4 to 128 dislocations. On the left is the whole curve, on the right a close-up from the drawn rectangle. The curve seems to be independent of size, except for the onset at 0-quantile (global minimum force) and for some wrinkles at quantiles 0.12, 0.50 and 0.88.*

The idea of using the predictive models is to find an optimal mapping (a linear combination in the simplest case) from the features to the targets. In the case of the quantiles, a feature that has by itself a small correlation with the target might turn out to be useful for the prediction if it contains new information that the other features don't have. For example, two very nearby quantile values contain much overlapping information since they are highly linearly dependent (correlated), leaving little room for predictability improvement when linearly combining them. In this work, from each chosen feature category, (usually 64) quantile features are chosen uniformly from the whole interval [0,1] as inputs for the predictive models.

In addition to the 4 pinning feature categories of Figure 19, the initial dislocation state is described with the values of the 4 fields (pinning energy, force, force derivative and force second derivative) at the initial, relaxed dislocation positions (x_j), presented as a sorted list that naturally contains the same number of quantiles as there are dislocations within the periodic loop. With these, another 4, experimental feature categories are used: the first three are derived from the local extrema of the pinning energy and force fields, and the fourth from the distances between relaxed dislocations. These last features are made with the intention to provide some information of how the field values or dislocations are

spatially related to each other, since this information is not directly contained within the first 8 feature categories (although derivative distributions tell something about local field shapes).

It is not certain whether the chosen features are the best. There could be other approaches that can provide features with better predictive power. The same predictive power could also be achieved with a different choice of features. There are some more feature choices tested on this work's datasets than just the presented ones, but there is no remarkable improvement in predictability when using those features. Some simple alternatives to quantiles, such as using only mean and standard deviation, are found to be insufficient when comparing the predictive power. Since the L^1-regularization method is used, the predictive models learn to only use important input features, and weight analysis (to be shown in Figure 23) can be used to see which feature categories are important and which not. It turns out that only a few of the presented 12 feature categories provide most of the predictive power.

4.4 Stress-Strain Curve Prediction

Stress-strain curves are first predicted from hand-picked features using a regularized linear regression model and variants of simple artificial neural networks (4.4.1), focusing on the 64 dislocation system. Weight analysis is performed on the linear regression model, showing the regularization penalty (assumed to correlate with contribution to prediction) of the chosen feature sets. Predictability obtained using the linear regression model is shown for different system sizes.

For the system of 16 dislocations, linear regression model results are compared with results of varying convolutional neural network models that mainly vary on the amount of input or output features (4.4.2). Convolutional networks are found to be challenging to use; penalty-type regularization causes the training to fail in most cases, and there is a clear performance difference when predicting the whole stress-strain curve instead of a stress at a single strain.

4.4.1 Stress Prediction Using Selected Features

Figure 21 shows the average correlation between true and predicted stresses over 5 training instances for the 64 dislocation system dataset using the models described in Table 1 and the 12 feature categories described in 4.3 as input (will be listed again in Figure 23). All models give roughly the same level of predictability: starting with almost perfect correlation at the lowest strains, ending with around 0.84 ± 0.01 correlation for the flow stress predictions. Some models occasionally have trouble at the lowest

stresses so that sometimes the model finds the perfect solution and sometimes a much worse solution. This is reflected in the standard deviation of the correlation, shown in the same Figure 21.

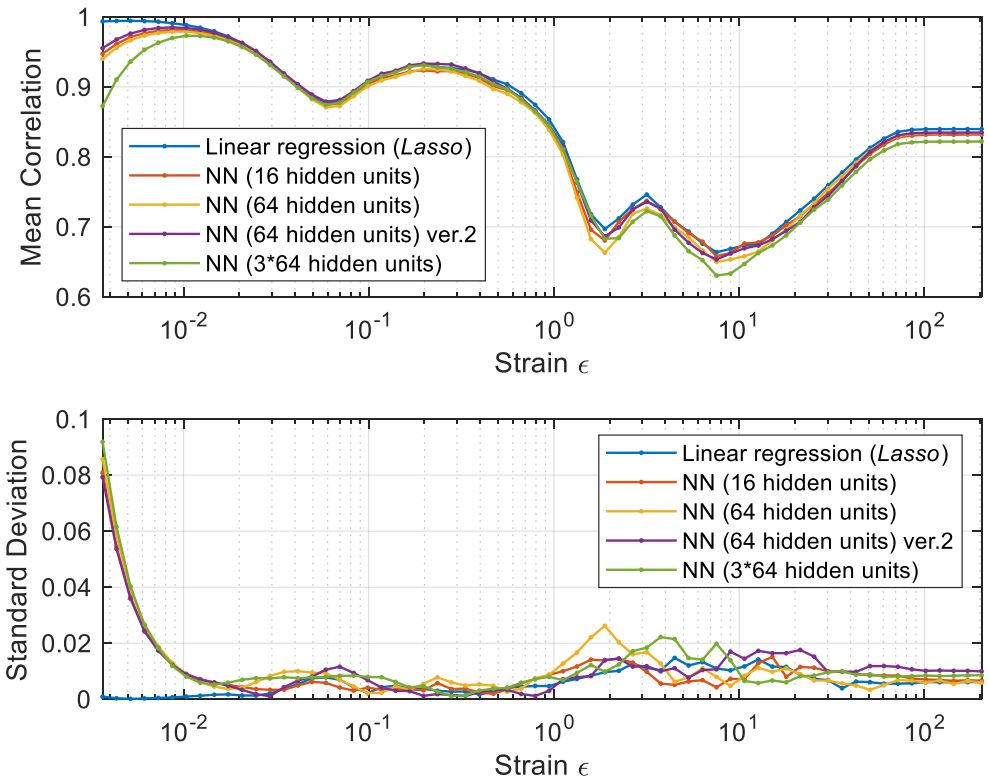

Figure 21. *Predictability of stresses at given strains (64 dislocation system), measured using the correlation between testing set target stresses and predicted stresses. The shown result (above) is a sample mean over 5 trained models, each using their own random division into training and testing sets. Also shown (below) is the sample standard deviation of the 5 correlation realizations at each strain. Different line colors indicate different types of models which are described more deliberately in Table 1.*

The neural network models have varying amounts of hidden units and regularization. One of the models has multiple *hidden layers*. As explained in 2.3.1, using more than one hidden layer in a simple neural network is not supposed to change the result, since a model with one intermediate result has full theoretical capability to produce any non-linear relation, given enough hidden units. However, this type of a model is sometimes used in other works [4][5], so it was decided to test it here in practice as well.

Since the best working model seems to be the L^1-regularized linear regression model *Lasso*, it seems that there are no notable non-linear relations between the chosen features and the targets, but the dependency appears to be just linear (mathematically *affine*). Figure 22 shows examples of some of the testing set target stress-strain curves with the predictions from the *Lasso* model, both original and normalized versions.

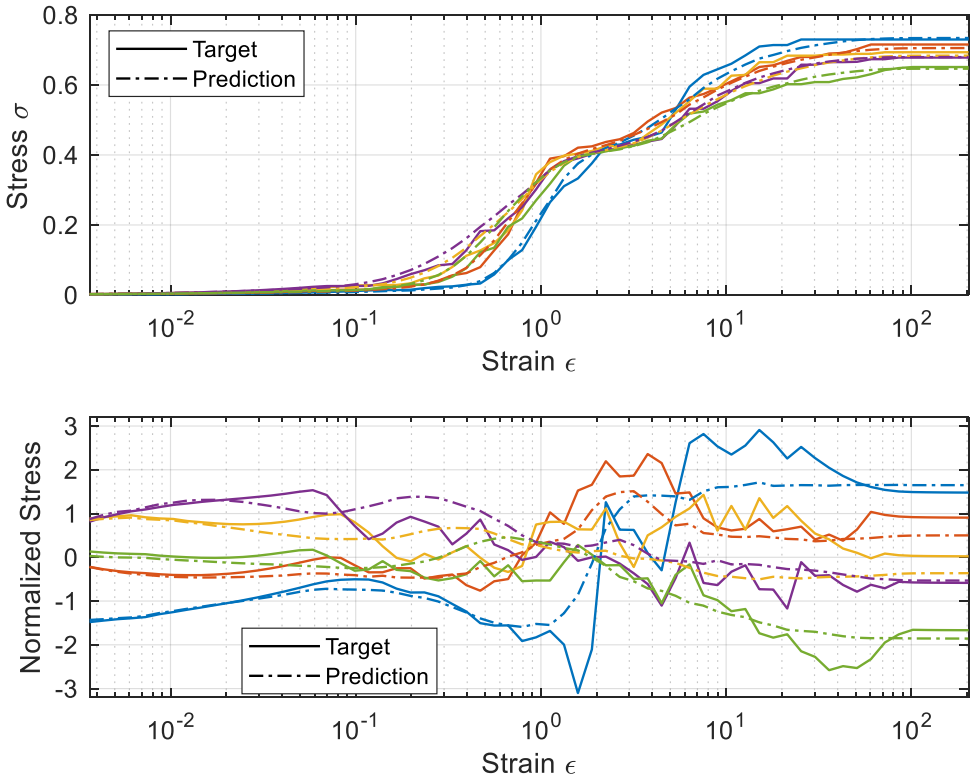

Figure 22. *Example stress-strain curves (64 dislocation system) with Lasso predictions (above), and corresponding normalized versions (below). These examples are from the testing set (not used in the training).*

Since the models utilize heavy L^1-regularization, analysis of the learned weights becomes straightforward. Features that contribute to predictability are allowed to have high regularization penalty (in exchange for improving predictability by decreasing *loss*), and vice versa for the unimportant features. Figure 23 shows how much penalty is generated by the weights corresponding to each feature category. At the start, the pinning force derivative at the relaxed dislocation positions, $\mathrm{D}F(x_j)$, is the most active feature, but also the pinning energy at the dislocation positions, $E(x_j)$, is used. This is expectable, since using the force derivatives corresponds to a linear estimation of how the simulation would begin. Use of the initial pinning energies at the middle strains is a bit less intuitive but seems to work to some extent. The middle section of the studied strain interval also includes transition zones where mixes of many features are used for the prediction. Flow stress prediction is done mainly with the quantiles from the pinning force (F) categories, both the original and the alternative with only the local extrema points.

All the previous results are for the system with 64 dislocations. Figure 24 shows how the predictability curve depends on the system size, expressed with the number of dislocations.

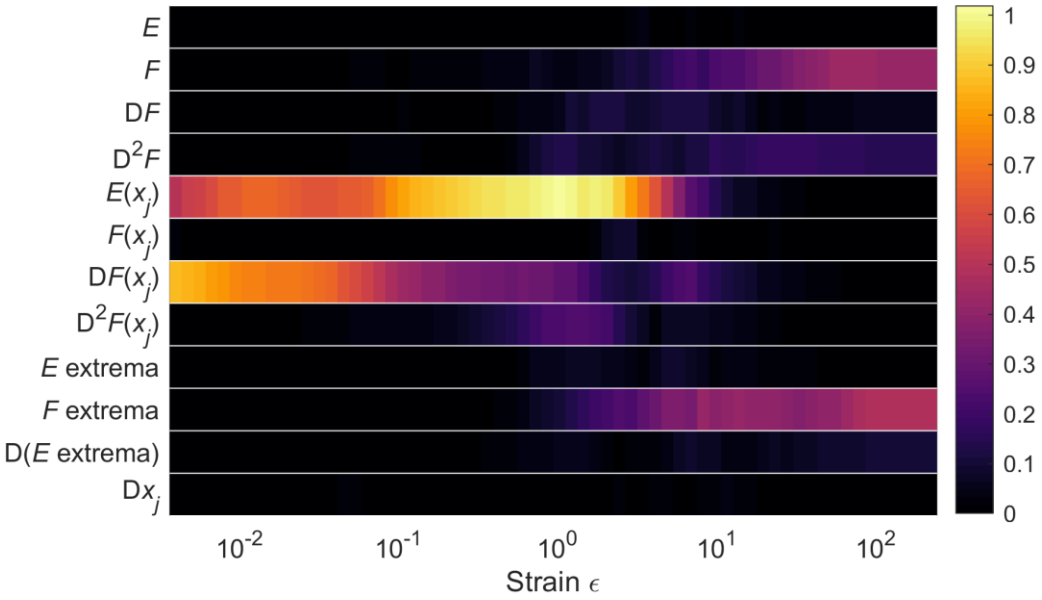

Figure 23. *Weight (category) analysis of the trained Lasso model for the 64 dislocation system. Color indicates the total L^1 penalty from each feature category's weights (each category contains 64 quantiles uniformly from the range [0,1]) as a function of strain, averaged over 5 training instances. Categories 1-4 (starting from the upmost) contain quantiles of the pinning energy, force, force derivative and force second derivative distributions. Categories 5-8 are quantiles of distributions of the same quantities as 1-4 but evaluated only at the relaxed dislocation positions (at the beginning of the stress-strain curve simulation). Categories 9-11 are quantiles from distributions derived from local extrema of the pinning fields, and category 12 contains the quantiles of the distances between neighboring relaxed dislocations. The symbol x_j refers to the relaxed dislocation positions in general.*

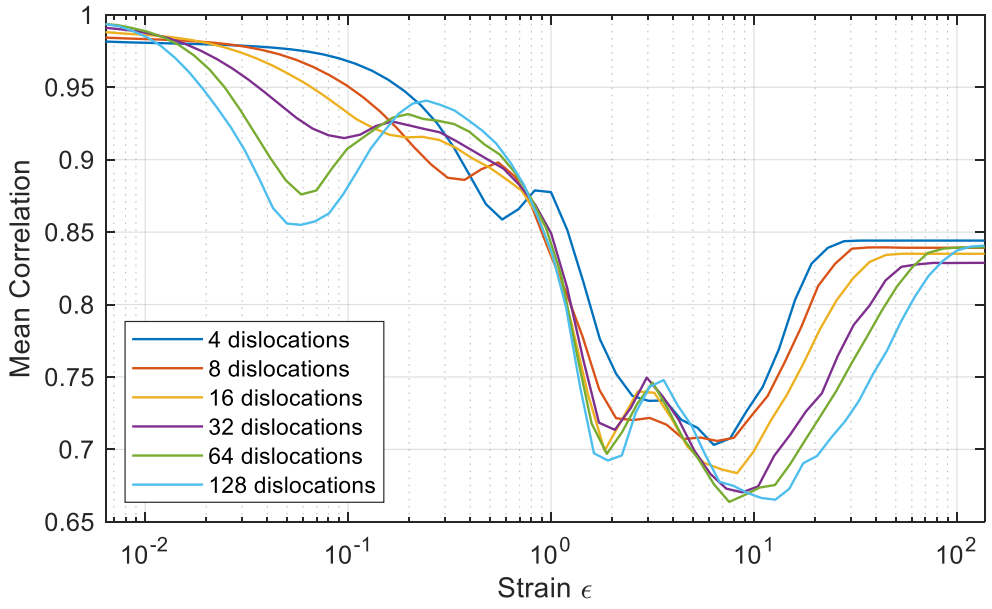

Figure 24. *Predictability of the stress-strain curve for different system sizes (Lasso model; mean correlation over 5 training instances).*

4.4.2 Stress Prediction Using A Convolutional Neural Network

The convolutional network model can find relations between spatial shapes of spatially defined features (channels) and the targets to be predicted. It can also be made shift-independent, like in the case of this work, where the studied systems are periodic. Figure 25 shows predictability curves from *Lasso* and various convolutional models for the system size of 16 dislocations. As explained in 3.2, using the convolutional network model poses some challenges not encountered when using the simpler models of 4.4.1. Predicting the whole stress-strain curve at once produces notably worse results than when training a model for each strain point individually. Using the L^1-regularization method improves predictability, but only when predicting flow stress alone. In other cases, using penalty-type regularization is found to halt the training, suppressing weights to 0.

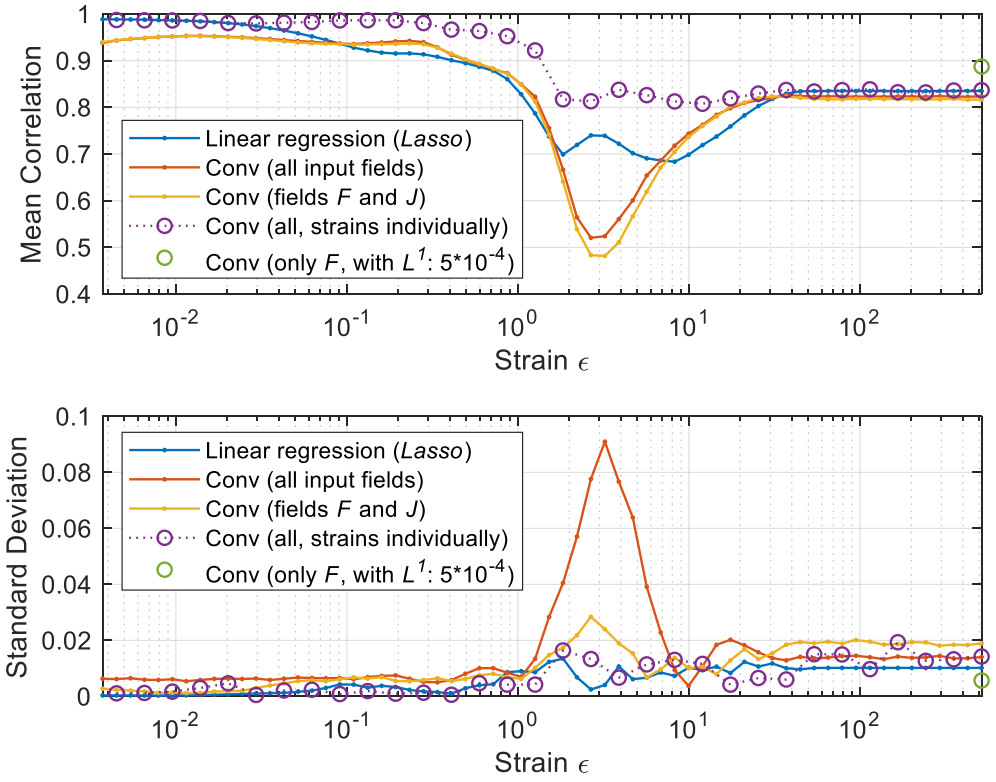

Figure 25. *Sample mean (above) and sample standard deviation (below) of correlation between the stresses and 5 prediction realizations of the stresses for each predictive model. The results are for the 16 dislocation system dataset. The first model is the Lasso from Figure 24, and the rest are variating convolutional models. Second and third model are otherwise the same but trained for a different number of input fields (all means E, F,* D*F,* D^2*F and J; J is the dislocation position indicating feature explained in 3.2, the rest are pinning fields). The fourth set of points is the result from a collection of models, where a separate model is trained for each strain point individually (using all the input feature channels, as the second model uses). The fifth set, a single point, is the result when predicting only flow stress using only the pinning force field F as input and with L^1-regularization penalty parameter $5*10^{-4}$ for all the convolutional weights (the other convolutional models are not regularized).*

The tested convolutional models all have the same structure, but they vary by the number of inputs and outputs, and by regularization in the one case. The first two convolutional models (red and yellow solid lines) in Figure 25 show that there is not much difference when giving the derivatives/integrals of the force field as an additional input, since a convolutional network has capability to calculate them from the force alone (for example, a convolutional operator $\mathbb{C}$ with a size 1×3 convolutional *kernel* [-a a 0], for some $a \in \mathbb{R}$, effectively differentiates a single-channel input field).

The best prediction result for the whole stress-strain curves is the line of connected circles in Figure 25, consisting of results from 160 separate models; each of the 32 circles represents an average over 5 training instances. As a small note, the convolutional networks have a small chance to fail. Two of the original 160 models fail to train (producing correlation near zero) near strain $\varepsilon \approx 10^{-1}$, but the failed instances are replaced by additional instances (161st and 162nd) to obtain the curve of connected circles in Figure 25.

The best flow stress prediction, with around 0.89 correlation, is from the regularized convolutional model focusing only on flow stress. The convolutional network method clearly has potential to exceed the linear regression model results, which is not unexpectable, since the convolutional model can process spatial information that is unavailable to the regression models. The chosen hand-picked features, given as input to the simpler models, don't contain much spatial information, at least not directly. Theoretically it should be possible to figure out features matching the relations the convolutional model learns, but the relations may be far from obvious.

With optimal choices of the training parameters and regularization, predictability might be improved further. The convolutional network model is however somewhat tricky to optimize, and it is computationally much heavier than the models of 4.4.1, unless parallel computing resources, such as specialized graphics processing units, GPUs, are available. Hopefully there is more automatization involved in future versions of machine learning interfaces, without leaving the parameter optimization work to every user, and maybe the training process could be optimized by somehow reducing unnecessary computational work so that models would learn more efficiently.

4.5 Avalanche Statistics

Avalanches (horizontal phases of the stress-strain curves, excluding the eternal flow stress avalanche) form their own statistics, which are visualized in the following Figures. First, Figure 26 shows the relation between system size and the average total number of avalanches. The relation between system size and the number of avalanches starts

off linear; a large system seems to contain approximately the avalanches that its smaller subsystems would have if it were to be divided. At larger systems, the relation starts to bend towards lower values than the linear relation would suggest. The bending could be caused by an effect where hypothetical subsystem avalanches at nearby stresses are joined, forming a single avalanche; a lower stress avalanche can trigger a hypothetical higher stress avalanche at the lower stress since dislocations can push each other forward. Even if some dislocations are far apart, they can push each other due to the long-range harmonic interaction between dislocations (see 2.1.2) that extends past immediate neighbors.

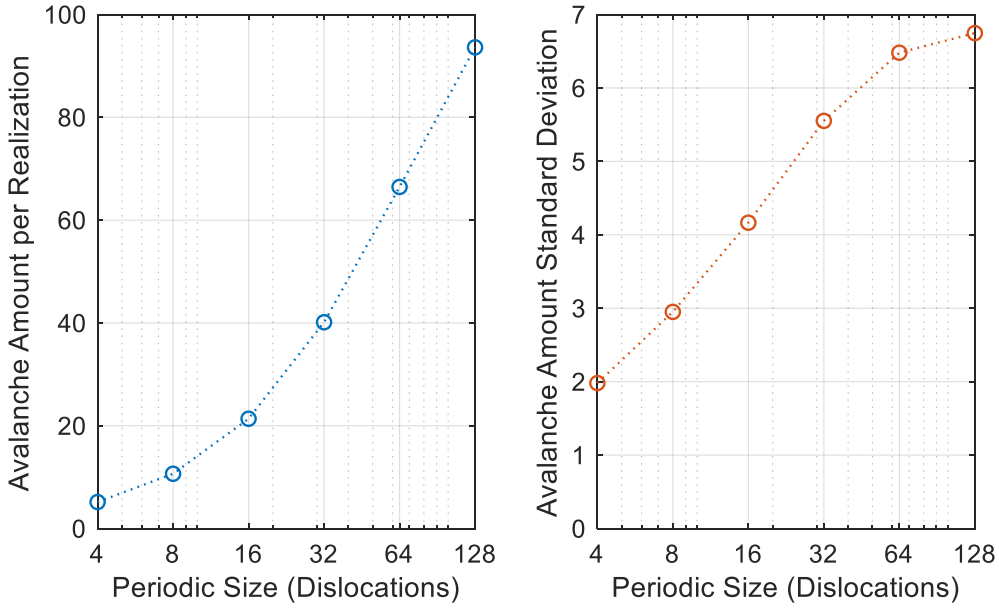

Figure 26. *Sample mean (left) and sample standard deviation (right) of the number of avalanches per a stress-strain curve realization as a function of the periodic system size.*

Figure 27 shows how avalanches distribute over stress, strain or avalanche size. The first one is computed as usual, while the other two are $\log_{10}$-densities, meaning density over $\log_{10}$-strain and $\log_{10}$-size (in analogy with the definition of the *log-normal distribution*). This means that during normalization, each histogram bin is divided by the logarithmic bin width instead of the linear width. This choice produces clearer distributions from the visual point of view.

Distributions over stress and ($\log_{10}$-)strain seem to have a limit shape towards which the distributions converge as system size is increased. They also have peaks at low stress and strain, which could be caused by metastability after relaxation, where even a small push would be enough to trigger movement. The separation of multiple peaks at small strains could be due to numerical accuracy (see technical details of how avalanches are

detected in 3.2). Local maxima at the centers of the stress and ($\log_{10}$-)strain distributions are most probably a consequence of the peaks in the force distribution (Figure 7).

Distribution over ($\log_{10}$-)size has a quite clear cut-off at the size of 16 strain units, which is the average distance between neighboring dislocations. When dislocations move more than this much, they could be considered to have replaced their neighbors, and therefore there is a high chance that a flow state has been reached. However, there is some leak (avalanches larger than 16 strain units), especially in larger systems. After moving 16 strain units on average, dislocations don't necessarily settle in the original configuration that would then cause a periodic cycle (flow state) to begin. Instead, the dislocations could go through new configurations where they have a chance to get stuck, explaining the leak effect in large systems.

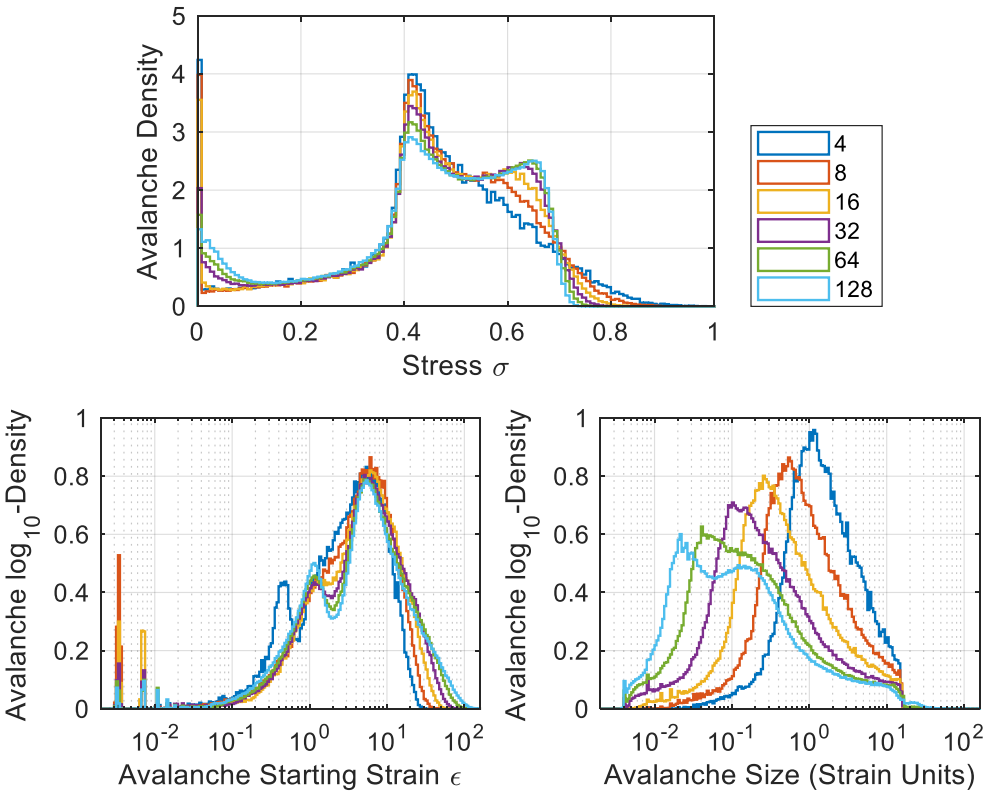

Figure 27. *Probability density distributions of the occurrence of an avalanche depending on stress (top),* $\log_{10}$*-strain (left) and* $\log_{10}$*-size (right) for periodic system sizes of 4…128 dislocations.*

Next, Figure 28 shows a bivariate avalanche distribution over stress and $\log_{10}$-size for the system of 64 dislocations. There is a leaf-shaped pattern which shows that the maximum avalanche size depends heavily on the stress. The monotonic relation is broken at special stress values, probably again caused by the oddities in the force distribution of

Figure 7. Minimum avalanche size doesn't seem to depend much on the stress, but certain avalanche sizes occur more often at certain stresses.

Theories suggest [1][3][6] that the probability of an avalanche decays as a function of size with a power law relation when near phase transition point (near flow stress) within some size range. (Probability over $\log_{10}$-size would also then have a power law, but not necessary with a negative power.) It has also been claimed [6] that critical events (such as avalanches) in general are difficult to predict, or even unpredictable. The first claim about power laws is not studied extensively in this work, but it is mentionable that the avalanche distributions derived from this work's datasets are not conflicting with the claim. The next subchapter attempts to test the claim about predictability.

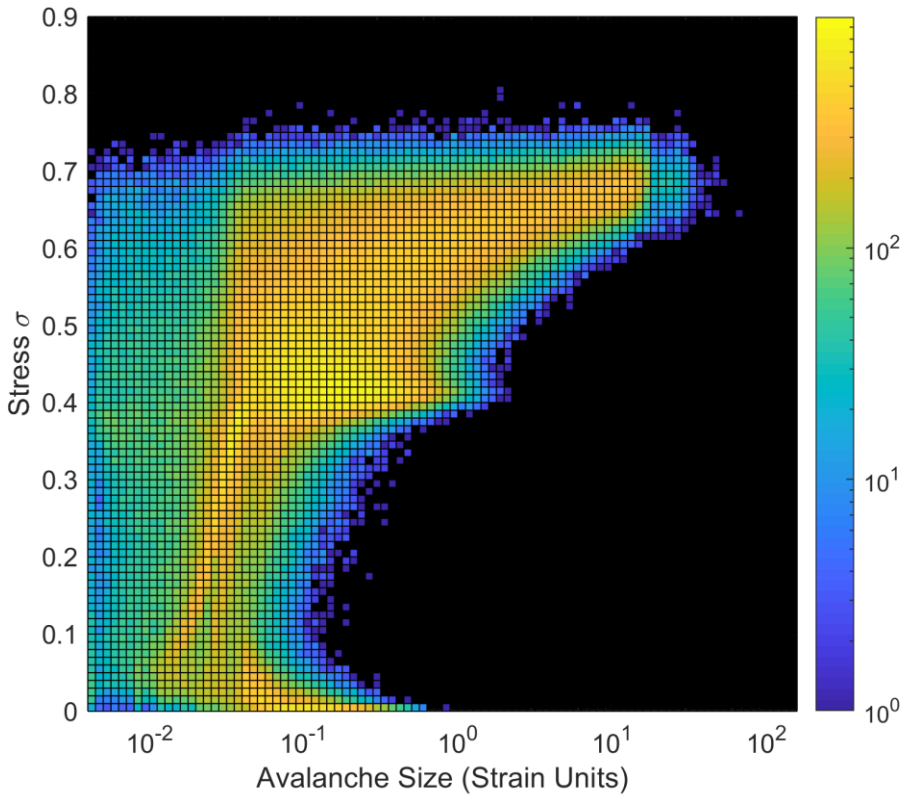

Figure 28. *Bivariate histogram of the avalanche count (color) depending on avalanche size and stress, from the dataset of the 64 dislocation system. The tiny square pixels are the histogram bins. Black areas don't contain any avalanches.*

4.6 Avalanche Prediction

Avalanche prediction starts with defining some property as the prediction target. The number of avalanches within a stress-strain curve is not constant, and there might not even be any avalanches (some such cases are found from the dataset of the 4 dislocation system). Predicting avalanches depending on strain could be troublesome, since a

strain value is mainly an integration (sum) over the past avalanche events, so failing to predict a low strain avalanche affects the prediction at higher strains. In this work, probability of avalanche occurrence (probability density), depending on stress and on the size of the avalanche, is chosen as the prediction target. Figure 29 shows three examples of the avalanche maps, and the generation of these maps is explained next.

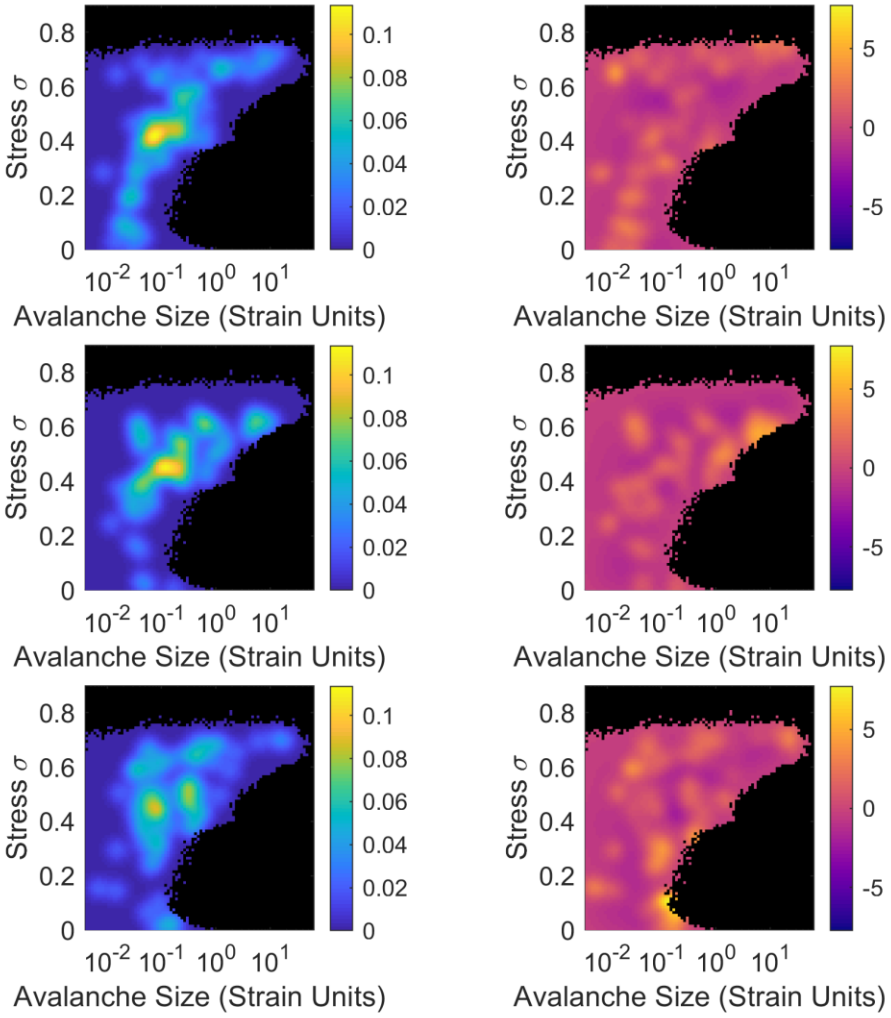

Figure 29. *Three example target avalanche maps, original (left) and normalized (right). A map is obtained by having a count histogram (like in Figure 28, but for a single realization) and blurring it with a normal distribution (standard deviation is chosen to be 3 pixels in both directions; one pixel has height of 0.01 stress units and width of 0.05* $\log_{10}$*-size units).*

Since avalanches are discrete events, they don't form any unique continuous density distribution, but instead the density needs to be defined artificially. If there were a large number of avalanches covering the whole studied range of stresses and sizes, a histogram would do, but the avalanches of a single stress-strain curve turn out to be quite

sparse within the size-stress space for the system sizes studied in this work. The target avalanche maps in this work are formed by generating a normal distribution at the avalanche points (residing in the size-stress space), and the avalanche map is the resulting sum. The same grid of stress and avalanche size bins are used as in Figure 28, but the definition is independent of the grid choice when using the same standard deviation parameters for the normal distributions. Note the use of logarithmic avalanche size. As an additional detail, areas where no avalanches occur (black regions of Figure 28) are left unaffected by the normal distribution blurring.

In practice, a map is presented to a predictive model as a vector of target features, with each feature corresponding to a bin of the two-dimensional grid. Prediction is done from the same input features as in stress prediction, explained in 4.3 (and listed compactly in Figure 23). Simple neural networks (64 hidden units) produce the predictability result of Figure 30, averaged over 5 training instances. The beak-like pattern has the highest predictability regions near the largest avalanches of each stress, especially at the lowest stresses, and also at a wide size range at the highest stresses.

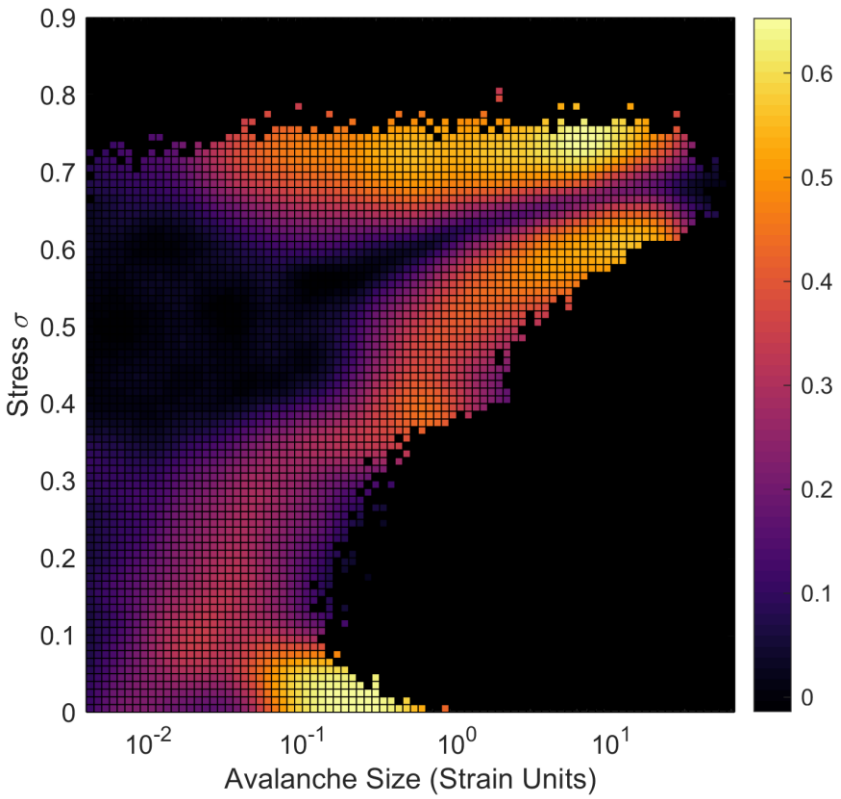

Figure 30. *Predictability of avalanche maps, measured with the mean testing set correlation (color) over 5 training instances. The model is a simple neural network (operator* LN*) of 64 hidden units trained with L^1 penalty parameter $1*10^{-5}$ for both weight matrices, predicting from the same features as in 4.4.1.*

The correlation is only about 0.65 at most, so the predictions are still far from perfect. Predicted maps of Figure 31, corresponding to the examples of Figure 29, show that the model can vaguely estimate the areas where avalanches are likely to appear more or less densely than in the average case. It can also be observed that the high stress avalanches of a wide size range may often appear as a bunch, covering a wide size region.

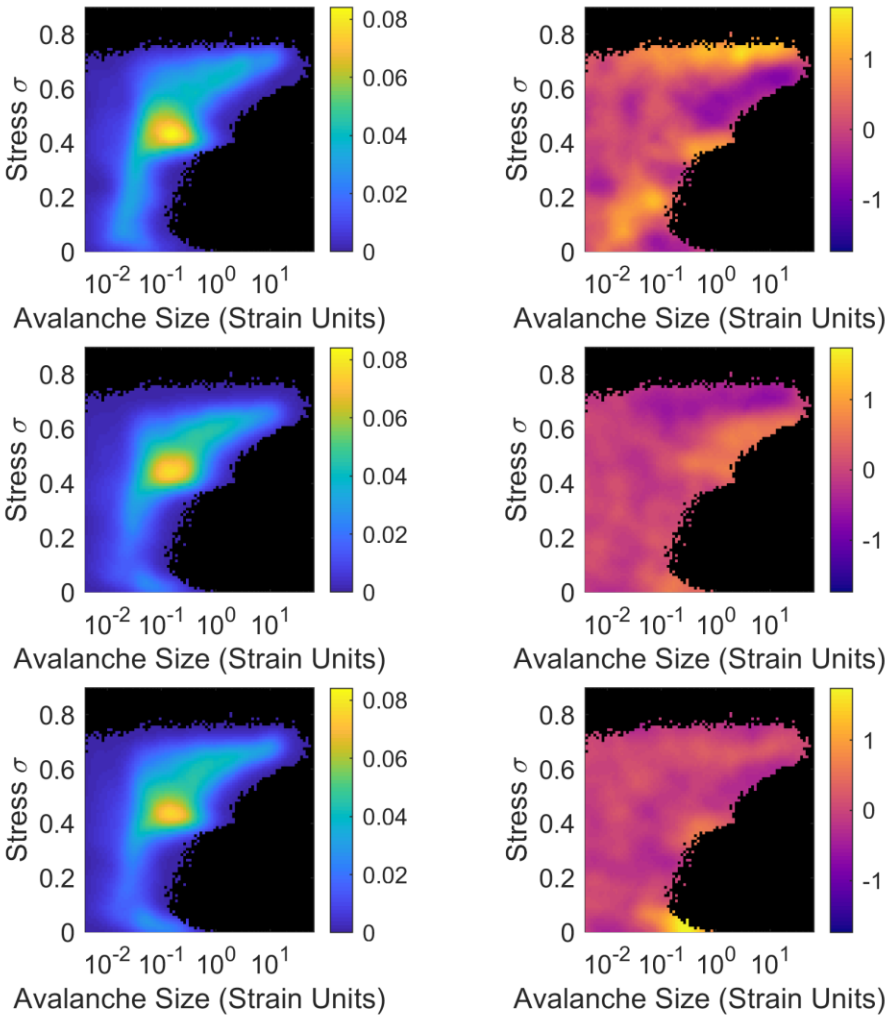

Figure 31. *Example predicted avalanche maps (corresponding to the targets of Figure 29). These examples are from the testing set (not used during the training). The predictive models always try to predict the normalized versions (shown at the right side).*

The area with low stresses but large avalanche sizes is quite highly predictable, probably due to the input features describing the initial state. Appearance of avalanches at the highest stress areas could be expected to correlate with the flow stress (which is required to be high enough if avalanches were to appear at the very top of the map), so the ability

to predict flow stress should help when predicting the top regions of the avalanche maps. In the middle however, there are regions with no predictability.

The convolutional network model could be powerful for this problem since avalanches depend heavily on the spatial details of the pinning fields. It could be expected that since predicting a flow stress corresponds to prediction of the eternal flow state avalanche, predictability of flow stress would indicate some upper limit for the predictability of non-flow avalanches as well. The studied avalanche prediction problem involves a large amount of targets to be predicted, and with the remarks of 4.4.2 about efficiency, convergence and optimization problems encountered with the convolutional models, it is decided to leave this direction (of studying convolutional network capability on this problem) outside the scope of this work.

5. CONCLUSION

Simulated stress-strain curves of the one-dimensional dislocation models are found to be quite predictable with simple regression models and convolutional networks, given that the initial dislocation states and the pinning landscapes are known. Predictions are not accurate enough that they would contain clear descriptions of individual avalanches, which is a similar result as in [4]. The avalanche prediction problem, where avalanche density maps are predicted, turns out to be difficult for the simple neural network model. Even so, it is found that some parts of the avalanche maps are predictable with relatively small but still noticeable correlation coefficient.

A convolutional network model can read spatially defined input channels but is computationally heavy and difficult to optimize through regularization. Simple regression models learn quickly and work well with L^1-regularization but require expressing the input quantities without direct spatial information. Quantiles are found to be efficient descriptors of the pinning fields, and the 5%-quantile of the pinning force field alone is found to have around 0.80 correlation with the flow stress, without dependence on the size of the periodic system. The simple regression models improve the flow stress predictability to about 0.84 by combining the information within the chosen input features. Simple neural networks are found not to improve the result of the regularized linear regression model. A convolutional network model can harness the spatial information lacking from the quantile description used by the simpler models to produce flow stress predictions with close to 0.89 correlation. Flow stress is therefore found to be quite, but not entirely, predictable using the studied predictive models. Also, flow stress is found to be very independent of the initial state of the dislocations.

Stress-strain curves contain phases that have varying degrees of predictability. As in a previous study about the predictability of a related system [4], the beginning of the stress-strain curve turns out nearly fully predictable. Stresses at the middle sections of the studied (log-)strain intervals have high differences in predictability between the predictive models and some differences between periodic system sizes. Weight analysis shows that the linear regression model learns to use different sets of input features depending on the phase of the stress-strain curve. Overall, the convolutional neural network model is found to be the most effective for predicting the stress-strain curves.

Avalanches, claimed to be unpredictable by nature [6], are found to be difficult to predict from the chosen features with a simple neural network model. Only certain avalanches

with a specific size at a given stress turn out to be slightly predictable. Totally unpredictable size-stress regions appear when the stress is high enough. The study could be continued by using convolutional neural networks to find out if the avalanche predictability could be improved, or if some avalanches are so unpredictable that even convolutional networks will not work. The avalanche prediction problem could also be defined alternatively in many different ways.

The obtained prediction results could be improved with more advanced learning techniques or through better optimized parameter choices. Maybe some of the remaining correlation is too difficult for the predictive models to achieve. It could be a consequence of the theory behind dislocation movement and avalanches, which could be studied further. Predictability research could be continued by studying other systems, such as more realistic ones with less assumptions. Predictability results from resembling systems, such as from the two-dimensional dislocation model studied in [4], from the earthquake model of [6], or from other possibly studied systems listed in [3], could be reviewed together to find similarities and differences.

REFERENCES

[1] P. Moretti, M-C. Miguel, M. Zaiser, S. Zapperi, Depinning transition of dislocation assemblies: Pileups and low-angle grain boundaries, Physical Review B - Condensed Matter and Materials Physics, 2004, 69 (21), art. no. 214103, pp. 214103-1-214103-11.

[2] F. Leoni, S. Zapperi, Dislocation mutual interactions mediated by mobile impurities and the conditions for plastic instabilities, Physical Review E - Statistical, Nonlinear, and Soft Matter Physics, 2014, 89 (2), art. no. 022403.

[3] J.P. Sethna, K.A. Dahmen, C.R. Myers, Crackling noise, Nature, 2001, 410(6825), 242-250.

[4] H. Salmenjoki, M. J. Alava, L. Laurson, Machine learning plastic deformation of crystals, Nature Communications, 2018, 9 (1), art. no. 5307. Available: https://doi.org/10.1038/s41467-018-07737-2

[5] Y. Miyazawa, F. Briffod, T. Shiraiwa, M. Enoki, Prediction of cyclic stress-strain property of steels by crystal plasticity simulations and machine learning, Materials, 2019, 12 (22), art. no. 3668. Available: https://doi.org/10.3390/ma12223668

[6] C.-K. Pun, S. Matin, W. Klein, H.Gould, Prediction in a driven-dissipative system displaying a continuous phase transition using machine learning, Physical Review E, 2020, 101 (2), art. no. 022102.

[7] D. Steinberger, H. Song, S. Sandfeld, Machine learning-based classification of dislocation microstructures, Frontiers in Materials, 2019, 6, art. no. 141. Available: https://doi.org/10.3389/fmats.2019.00141

[8] D.C. Pagan, T.Q. Phan, J.S. Weaver, A.R. Benson, A.J. Beaudoin, Unsupervised learning of dislocation motion, Acta Materialia, 2019, 181, pp. 510-518.

[9] F. Lacombe, S. Zapperi, H.J. Herrmann, Force fluctuation in a driven elastic chain, Physical Review B - Condensed Matter and Materials Physics, 2001, 63 (10), art. no. 104104, pp. 1041041-1041047.

[10] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, 2012, 2, pp. 1097-1105.

[11] H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, Proceedings of the IEEE International Conference on Computer Vision, ICCV 2015, art. no. 7410535, pp. 1520-1528.

[12] (Website) JEOL Ltd. Glossary of Transmission Electron Microscope Terms. https://www.jeol.co.jp/en/words/emterms/search_result.html?keyword=dislocation (as of April 2020)